NEW HORIZON
COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

Department of Electronics and Communication Engineering

VI Semester

25NHOP610 – 5G MOBILE COMMUNICATION

LABORATORY MANUAL

Academic Year : 2025 – 2026
Semester : EVEN

NEW HORIZON COLLEGE OF ENGINEERING

Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

Department Vision

To create high quality engineering professionals who can serve the society and earn global recognition.

Department Mission

- To build strong foundation in Electronics and Communication Engineering aspects by exposing students to state of the art technology and research.
- To strengthen the curriculum through interaction with industry experts to equip the students with the required competency.
- To mould students to share technical knowledge and to practice professional and moral values.

Program Educational Objectives (PEOs)

PEO 1: To produce graduates with understanding of fundamentals and applications of Electronics and Communication Engineering.

PEO 2: To hone graduates with ability to apply, analyze, design and develop electronic systems.

PEO 3: To enhance graduates with latest technologies to enable them to engineer products for real world problems.

PEO 4: To build leadership qualities, management skills, communication skills, moral values, team spirit and lifelong learning ability for the graduates.

Program Outcomes (POs)

B. E graduate should possess the following Program Outcomes.

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems in Electronics and Communication Engineering.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems in Electronics and Communication Engineering reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes of Electronics and Communication Engineering that

meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge andresearch methods including design of experiments in Electronics and Communication Engineering, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities in Electronics and Communication Engineering with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice in Electronics and Communication Engineering.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions of Electronics and Communication Engineering in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

Program Specific Outcomes	
PSO1	To demonstrate the ability to design and develop complex systems in the areas of next generation Communication Systems, IoT based Embedded Systems, Advanced Signal and Image Processing, latest Semiconductor technologies, RF and Power Systems
PSO2	To demonstrate the ability to solve complex Electronics and Communication Engineering problems using latest hardware and software tools along with analytical skills to contribute to useful, frugal and eco-friendly solutions.

NEW HORIZON
COLLEGE OF ENGINEERING
New Horizon Knowledge Park, Ring Road, Marathalli
Autonomous College Permanently Affiliated to VTU, Approved by AICTE & UGC
Accredited by NAAC with 'A' Grade, Accredited by NBA

Department of Electronics and Communication Engineering

5G MOBILE COMMUNICATION

Course Code	**: 25NHOP610**	**Credits**	**: 3**
L:T:P:S	**: 3:0:0:0**	**CIE Marks**	**: 50**
Exam Hours	**: 03**	**SEE Marks**	**: 50**

LIST OF EXPERIMENTS

1. Write a MATLAB program to pass the Signal through a Rayleigh Channel.
2. Write a MATLAB program to pass the Signal through a Rician Channel.
3. Write a MATLAB program to generate the signal S(n) = 2 * n * (0.8^ n) corrupted by the noise d($\boldsymbol{n}$) resulting the signal $\boldsymbol{X}(\boldsymbol{n})$, $\boldsymbol{X}(\boldsymbol{n})$ = S(n) + d($\boldsymbol{n}$).
4. Write a MATLAB program to generate an exponential Sequence.
   $X(n) = (a)^n$ *for* $(i)\,0 \le a \le 1$ $(ii)\,-1 \le a \le 0$ $(iii)\,a \le -1$ $(iv)\,a > 1$
5. Demonstrate the downlink data transmission between UE and external data network (Ext-dn) using iperf command in OAI platform.
6. Demonstrate the uplink data transmission between UE and external data network (Ext-dn) using iperf command in OAI platform.
7. Write a MATLAB program to implement 2x2 MIMO.
8. Write a MATLAB program to compute the array factor for a ULA steered towards a desired angle and prints the coherent combining SNR gain.
9. Write a MATLAB program to generate NR-like OFDM baseband waveform and plot its PSD.
10. Write a MATLAB program to build a simple resource grid with PRB allocation and generate a waveform to display grid heatmap and spectrogram.
11. Write a MATLAB program to plot 3GPP TR 38.901 urban microcell (Umi) LOS and NLOS path loss curves versus distance.
12. Write a MATLAB program for: 5G URLLC for Industrial Automation.
13. Write a MATLAB program that simulates network slicing in 5G, where different types of services (e.g., eMBB, URLLC, mMTC) share the same physical infrastructure but are logically separated based on their QoS requirements.
14. Write a MATLAB program for RB scheduler with URLLC pre-emption and proportional sharing for eMBB/mMTC to simulate arrival and backlog over time.
15. Write a MATLAB program for simple IOT data transmission.
16. Write a MATLAB program to estimate user throughput for Carrier aggregation with two components carriers (CA_2CC).
17. Demonstrate the creation of pcap file, reading of pcap file and packet truncation with variable snap length.
18. Demonstrate the packet capturing and analysis using WIRESHARK tool by filtering the TCP and UDP packets.

LAB-IN-CHARGE **HOD-ECE**

Expt. No.	Name of the Experiment
1	Write a MATLAB program to pass the Signal through a Rayleigh Channel.
2	Write a MATLAB program to pass the Signal through a Rician Channel.
3	Write a MATLAB program to generate the signal S(n) = 2 * n * (0.8^ n) corrupted by the noise d($\boldsymbol{n}$) resulting the signal $\boldsymbol{X}(\boldsymbol{n})$, $\boldsymbol{X}(\boldsymbol{n})$ = S(n) + d($\boldsymbol{n}$).
4	Write a MATLAB program to generate an exponential Sequence. $X(n) = (a)^n \quad for \quad (i)\,0 \le a \le 1 \quad (ii)\,-1 \le a \le 0 \quad (iii)\,a \le -1 \quad (iv)\,a > 1$
5	Demonstrate the downlink data transmission between UE and external data network (Ext-dn) using iperf command in OAI platform.
6	Demonstrate the uplink data transmission between UE and external data network (Ext-dn) using iperf command in OAI platform.
7	Write a MATLAB program to implement 2x2 MIMO.
8	Write a MATLAB program to compute the array factor for a ULA steered towards a desired angle and prints the coherent combining SNR gain.
9	Write a MATLAB program to generate NR-like OFDM baseband waveform and plot its PSD.
10	Write a MATLAB program to build a simple resource grid with PRB allocation and generate a waveform to display grid heatmap and spectrogram.
11	Write a MATLAB program to plot 3GPP TR 38.901 urban microcell (Umi) LOS and NLOS path loss curves versus distance.
12	Write a MATLAB program for: 5G URLLC for Industrial Automation.
13	Write a MATLAB program that simulates network slicing in 5G, where different types of services (e.g., eMBB, URLLC, mMTC) share the same physical infrastructure but are logically separated based on their QoS requirements.
14	Write a MATLAB program for RB scheduler with URLLC pre-emption and proportional sharing for eMBB/mMTC to simulate arrival and backlog over time.
15	Write a MATLAB program for simple IOT data transmission.
16	Write a MATLAB program to estimate user throughput for Carrier aggregation with two components carriers (CA_2CC).
17	Demonstrate the creation of pcap file, reading of pcap file and packet truncation with variable snap length.
18	Demonstrate the packet capturing and analysis using WIRESHARK tool by filtering the TCP and UDP packets.

Prepared By	Verified By	Approved By
Ms. Lipsa Dash	**Dr. Pragati Tripathi**	**Dr. Aravinda K**
Senior Assistant Professor / ECE	**Senior Assistant Professor / ECE**	**Professor / HoD / ECE**

EXPERIMENT: 1

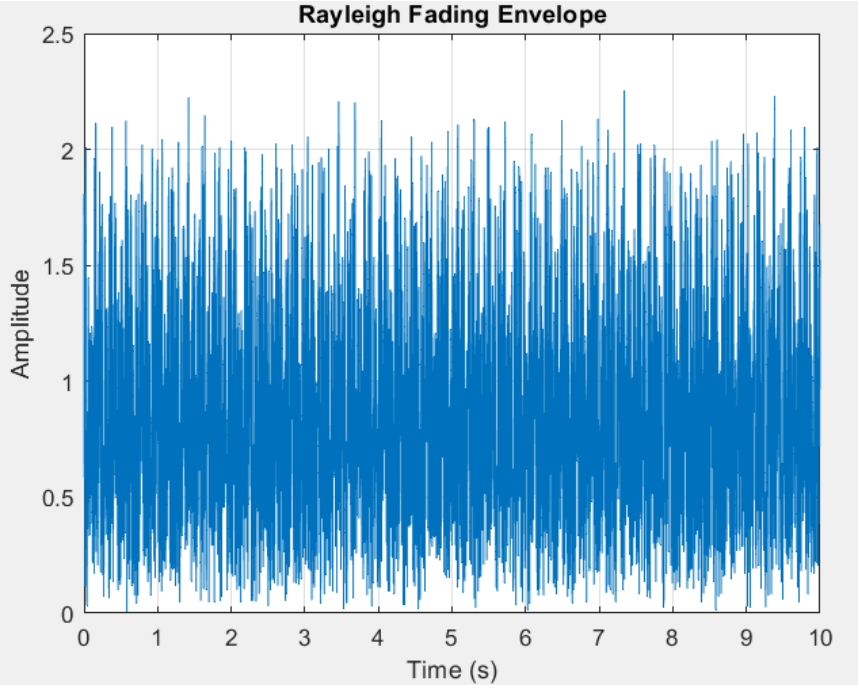
AIM: Write a MATLAB program to pass the Signal through a Rayleigh Channel.

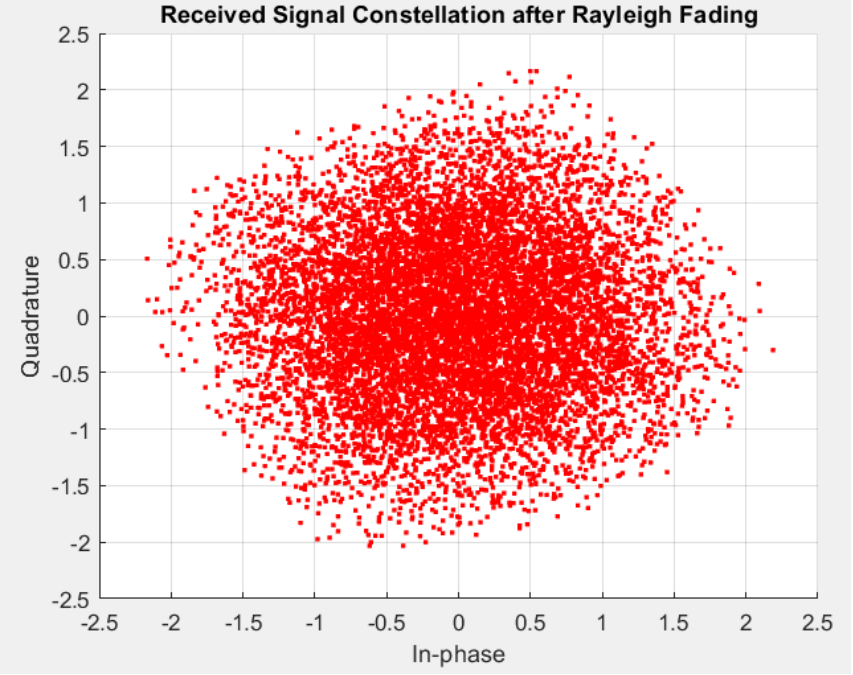
TOOL: MATLAB 2016

PROGRAM:

```
clc;
clear;
close all;

%% Parameters
N = 10000;      % Number of samples
fd = 100;       % Doppler frequency in Hz
fs = 1000;      % Sampling frequency in Hz
Ts = 1/fs;      % Sampling period
t = (0:N-1) * Ts;       % Time vector

%% Rayleigh Channel Model (Jakes' Model Approximation)
M = 20; % Number of sinusoids
theta = 2 * pi * rand(1, M); % Random initial phase
alpha = (2 * pi * (1:M)) / M; % Angles of arrival

% Generate Rayleigh fading process
rayleigh_channel = sqrt(1/M) * sum(exp(1j * (2 * pi * fd * cos(alpha)' * t + theta')), 1);
figure;
plot(t, abs(rayleigh_channel));
xlabel('Time (s)');
ylabel('Amplitude');
title('Rayleigh Fading Envelope');
grid on;

%% Simulated Communication Signal (QPSK)
data = randi([0 3], 1, N); % Random QPSK symbols
modulated_signal = exp(1j * (pi/4 + data * pi/2)); % QPSK modulation

% Pass through Rayleigh Channel
received_signal = rayleigh_channel .* modulated_signal;

%% Constellation Diagram
figure;
scatter(real(received_signal), imag(received_signal), 'r.'); xlabel('In-phase');
ylabel('Quadrature');
title('Received Signal Constellation after Rayleigh Fading');
grid on;
```

FIGURE WINDOW:





EXPERIMENT: 2

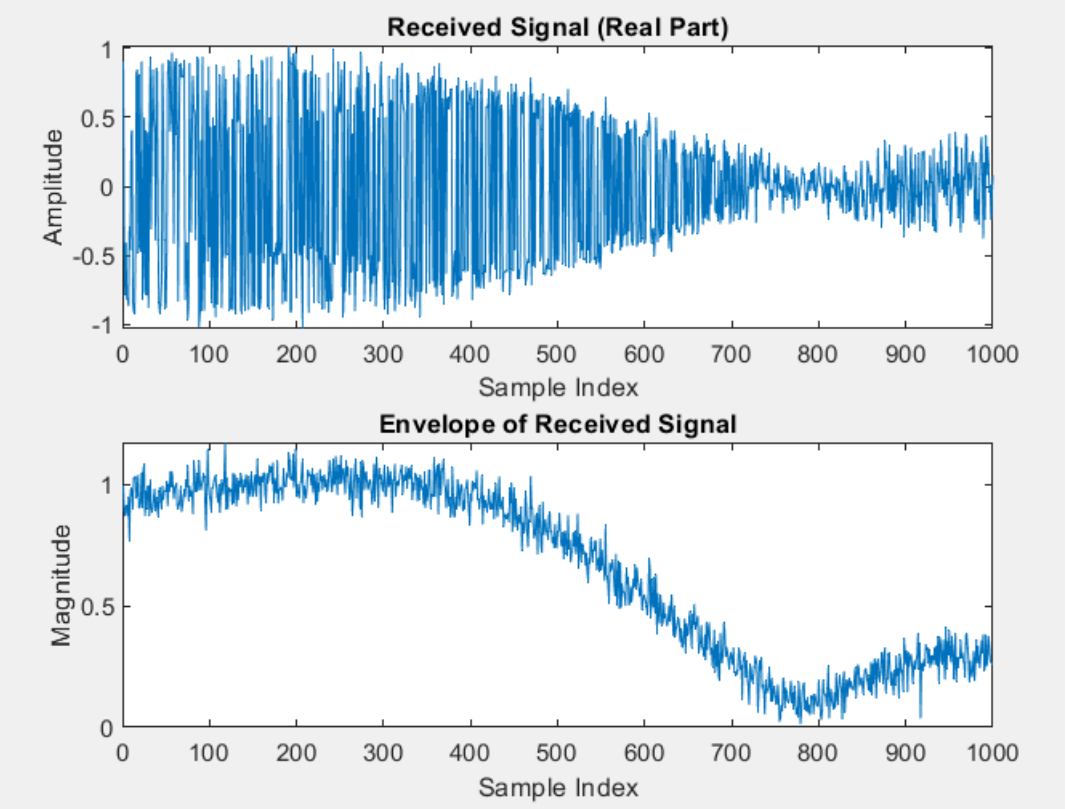
AIM: Write a MATLAB program to pass the Signal through a Rician Channel.

TOOL: MATLAB 2016

PROGRAM:

```
clc;
clear;
%% Parameters
fs = 1e6;        % Sampling frequency (Hz)
Ts = 1/fs;       % Sampling period
N = 10000;       % Number of samples
fc = 2e9;        % Carrier frequency (Hz)
fd = 100;        % Maximum Doppler shift (Hz)
K = 5;           % Rician K-factor (linear scale)
snr_dB = 20;     % Signal-to-noise ratio in dB

%% Time vector
t = (0:N-1)*Ts;

%% Rician Channel Generation using MATLAB built-in function
ricianChan = comm.RicianChannel(... 'SampleRate', fs, ...
'PathDelays', 0, ...
'AveragePathGains', 0, ... 'KFactor', K, ... 'MaximumDopplerShift', fd, ...
'DopplerSpectrum', doppler('Jakes'), ... 'RandomStream', 'mt19937ar with seed', ... 'Seed', 22, ...
'NormalizePathGains', true);

%% Transmit signal (random QPSK for demo)
data = randi([0 3], N, 1);
tx = pskmod(data, 4, pi/4);

%% Pass signal through Rician channel
rx = ricianChan(tx);

%% Add AWGN noise
rx_noisy = awgn(rx, snr_dB, 'measured');

%% Plot
figure;
subplot(2,1,1);
plot(real(rx_noisy(1:1000)));
title('Received Signal (Real Part)');
xlabel('Sample Index');
ylabel('Amplitude');

subplot(2,1,2);
plot(abs(rx_noisy(1:1000)));
title('Envelope of Received Signal');
xlabel('Sample Index');
ylabel('Magnitude');
disp('Rician fading simulation completed.');
```

FIGURE WINDOW:



EXPERIMENT: 3

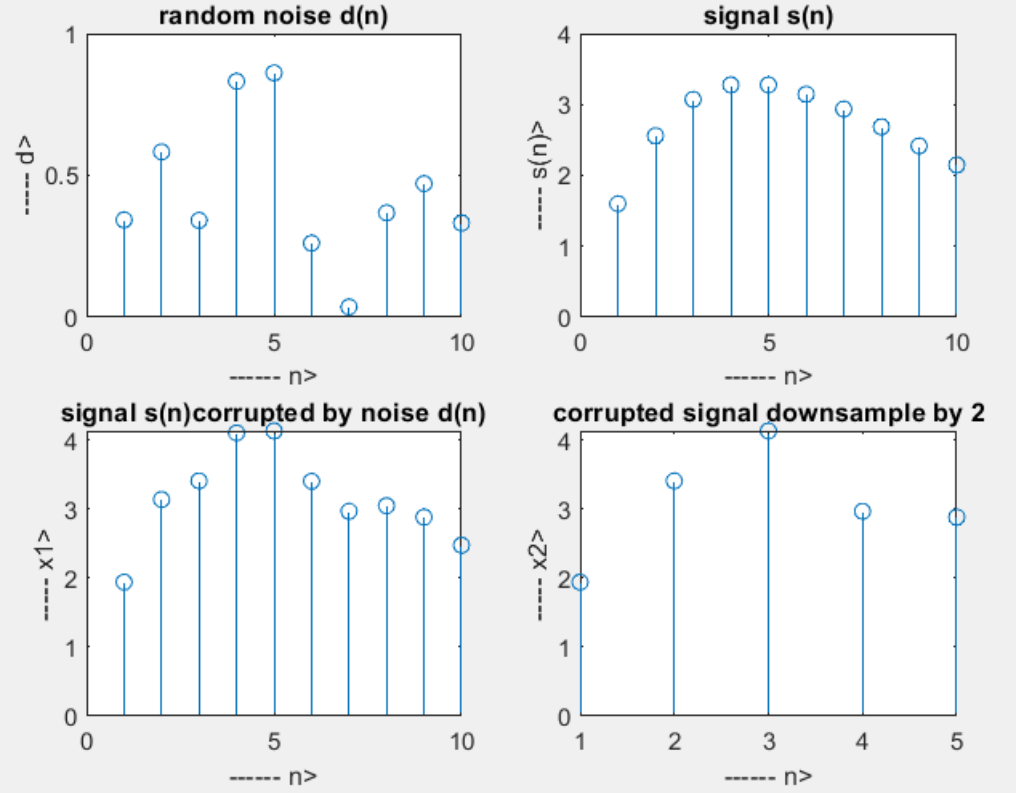
AIM: Write a MATLAB program to generate the signal S(n) = 2 * n * (0.8^ n) corrupted by the noise d($\boldsymbol{n}$) resulting the signal $\boldsymbol{X}(\boldsymbol{n})$, $\boldsymbol{X}(\boldsymbol{n})$ = S(n) + d($\boldsymbol{n}$).

TOOL: MATLAB 2016

PROGRAM:

```
clc;
clear all;
close all;
d=rand(1,10);
for n=1:10
s(n)=2.*n.*((.8)^n);
x1(n)=s(n)+d(n);
end

subplot(2,2,1);
stem(d);
xlabel('------ n >');
ylabel('----- d  >');
title('random noise d(n)');

subplot(2,2,2);
stem(s);
xlabel('------ n >');
ylabel('----- s(n)         >');
title('signal s(n)');

subplot(2,2,3);
stem(x1);
xlabel('------ n >');
ylabel('----- x1>');
title('signal s(n)corrupted by noise d(n)');

n=1:5;
for i=1:5
x2(i)=x1((2*i-1));
end

subplot(2,2,4)
stem(n,x2);
xlabel('------ n >');
ylabel('----- x2>');
title('corrupted signal downsample by 2');
```

FIGURE WINDOW:



EXPERIMENT: 4

AIM: Write a MATLAB program to generate an exponential Sequence.

$$X(n) = (a)^n \quad for \quad (i)\,0 \le a \le 1 \quad (ii)\,-1 \le a \le 0 \quad (iii)\,a \le -1 \quad (iv)\,a > 1$$

TOOL: MATLAB 2016

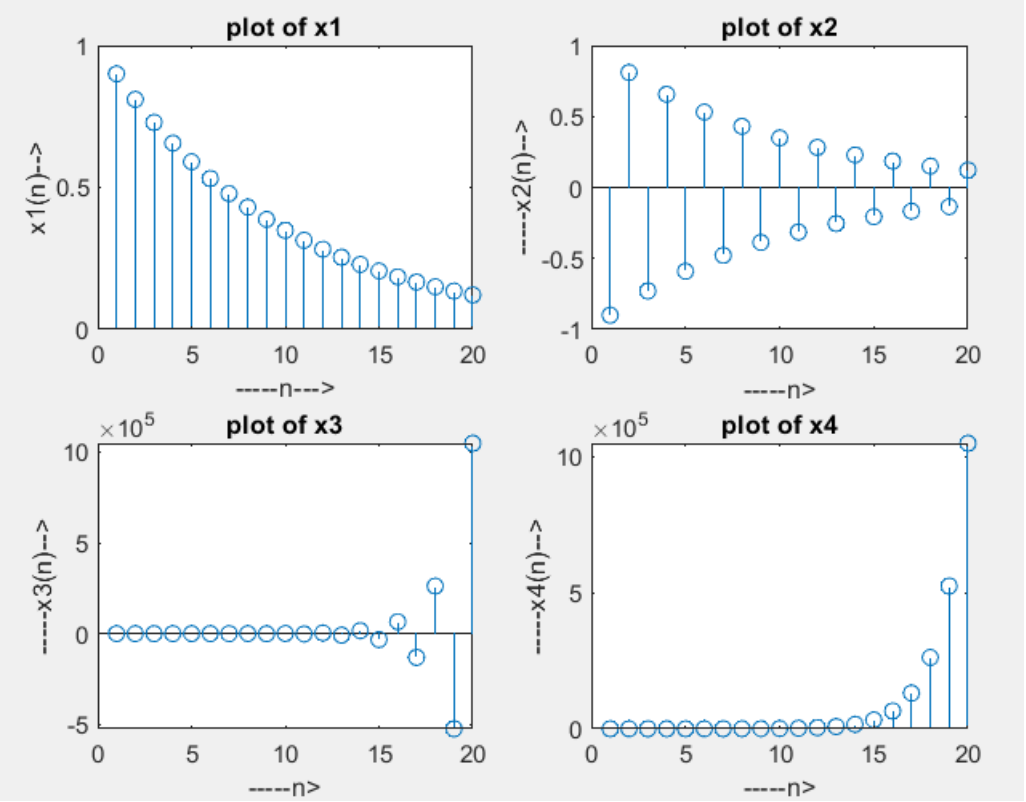
PROGRAM:

```
clc;
close all;
clear all;
a1=input('enter the value of a such that 0<=a<=1');
a2=input('enter the value of a such that -1<=a<=0');
a3=input('enter the value of a such that a<=-1');
a4=input('enter the value of a such that a>1');

for n=1:20
x1(n)=(a1)^n;
x2(n)=(a2)^n;
x3(n)=(a3)^n;
x4(n)=(a4)^n;
end

subplot(2,2,1)
stem(x1);
xlabel('-----n--->');
ylabel('x1(n)-->');
title('plot of x1');

subplot(2,2,2)
stem(x2);
xlabel('-----n   >');
ylabel('-----x2(n)-->');
title('plot of x2');

subplot(2,2,3)
stem(x3);
xlabel('-----n   >');
ylabel('-----x3(n)-->');
title('plot of x3');

subplot(2,2,4) stem(x4);
xlabel('-----n   >');
ylabel('-----x4(n)-->');
title('plot of x4');
```

COMMAND WINDOW:

enter the value of a such that 0<=a<=1
0.9
enter the value of a such that -1<=a<=0
-0.9
enter the value of a such that a<=-1
-2
enter the value of a such that a>1
2

FIGURE WINDOW:



EXPERIMENT: 5

AIM: Demonstrate the downlink data transmission between UE and external data network (Ext-dn) using iperf command in OAI platform.

H/W Requirement: Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz, 16 CPU, 32 GB RAM

S/W Requirement: Ubuntu 18.04 LTS, Docker latest version, docker-compose (1.29.2)

CODE:

```
sudo -i
cd /home
ls
cd openairinterface5g/ci-scripts/yaml_files/5g_f1_rfsimulator/
ls
docker-compose ps -a
docker-compose down
docker-compose up -d mysql oai-nrf oai-amf oai-smf oai-spgwu oai-ext-dn
docker-compose ps -a
watch docker-compose ps -a
Ctrl C
docker-compose up -d oai-cu docker-compose ps -a
docker-compose up -d oai-du watch docker-compose ps -a
Ctrl C
docker-compose up -d oai-nr-ue docker-compose ps -a
docker exec -it rfsim5g-oai-nr-ue /bin/bash
ping -I oaitun_ue1 -c 10 www.lemode.fr
ping  -I  oaitun_ue1 -c 2 192.168.72.135
Alt Ctrl T sudo -i
docker exec -it rfsim5g-oai-ext-dn /bin/bash
```

On UE : iperf -B 12.1.1.2 -u -i 1 -s

On ext-dn: iperf -c 12.1.1.2 -u -i 1 -t 20 -b 500K

EXPERIMENT: 6

AIM: Demonstrate the uplink data transmission between UE and external data network (Ext-dn) using iperf command in OAI platform.

H/W Requirement: Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz, 16 CPU, 32 GB RAM

S/W Requirement: Ubuntu 18.04 LTS, Docker latest version, docker-compose (1.29.2)

CODE:

```
sudo -i
cd /home
ls
cd openairinterface5g/ci-scripts/yaml_files/5g_f1_rfsimulator/
ls
docker-compose ps -a
docker-compose down
docker-compose up -d mysql oai-nrf oai-amf oai-smf oai-spgwu oai-ext-dn
docker-compose ps -a
watch docker-compose ps -a
Ctrl C
docker-compose up -d oai-cu
docker-compose ps -a
docker-compose up -d oai-du
watch docker-compose ps -a
Ctrl C
docker-compose up -d oai-nr-ue
docker-compose ps -a
docker exec -it rfsim5g-oai-nr-ue /bin/bash
ping -I oaitun_ue1 -c 10 www.lemode.fr
```

ping -I oaitun_ue1 -c 2 192.168.72.135

Alt Ctrl T

sudo -i

docker exec -it rfsim5g-oai-ext-dn /bin/bash

On ext-dn: iperf -B 192.168.72.135 -u -i 1 -s

On UE: iperf -c 192.168.72.135 -B 12.1.1.2 -u -i 1 -t 20 -b 500K

EXPERIMENT: 7

AIM: Write a MATLAB program to implement 2x2 MIMO.

TOOL: MATLAB 2016

PROGRAM:

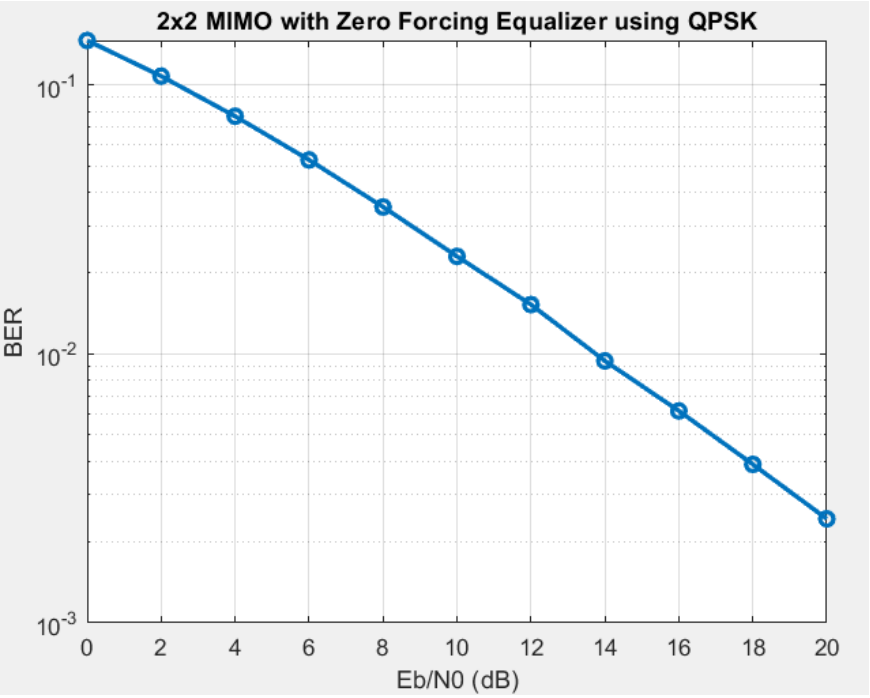
```
clc;
clear;

% Parameters
numSymbols = 1e5;      % Number of channel uses
M = 4;                 % QPSK
k = log2(M);           % Bits per symbol
EbN0_dB = 0:2:20;
nTx = 2;
nRx = 2;

% QPSK Modulator / Demodulator
modObj = comm.QPSKModulator('BitInput', true);
demodObj = comm.QPSKDemodulator('BitOutput', true);
ber = zeros(size(EbN0_dB));

for idx = 1:length(EbN0_dB)
   % Generate random bits for both transmit antennas
   bits = randi([0 1], numSymbols * nTx * k, 1);

   % Modulate bits to QPSK symbols
   symbols = step(modObj, bits);

   % Arrange symbols for 2 transmit antennas
   txSymbols = reshape(symbols, nTx, []);   % size = 2 x numSymbols
   N = size(txSymbols, 2);

   % Generate flat Rayleigh fading channel
   H = (randn(nRx, nTx, N) + 1j*randn(nRx, nTx, N)) / sqrt(2);

   % Noise variance
   EbN0 = 10^(EbN0_dB(idx)/10);
   noiseVar = 1 / (2 * k * EbN0);
   noise = sqrt(noiseVar) * (randn(nRx, N) + 1j*randn(nRx, N));

   % Received signal
   y = zeros(nRx, N);
   rxSymbols = zeros(nTx, N);

  for i = 1:N
      % Channel transmission
      y(:, i) = H(:, :, i) * txSymbols(:, i) + noise(:, i);
```

```
        % Zero Forcing Equalizer
        W_zf = pinv(H(:, :, i));     % ZF equalizer
        rxSymbols(:, i) = W_zf * y(:, i);
    end

    % Convert received symbols into serial stream
    rxSymbolsSerial = rxSymbols(:);

    % Demodulate
    rxBits = step(demodObj, rxSymbolsSerial);

    % BER calculation
    ber(idx) = sum(rxBits ~= bits) / length(bits);
    fprintf('Eb/N0 = %d dB, BER = %.4e\n', EbN0_dB(idx), ber(idx));
end

% Plot BER curve
figure;
semilogy(EbN0_dB, ber, '-o', 'LineWidth', 2);
xlabel('Eb/N0 (dB)');
ylabel('BER');
title('2x2 MIMO with Zero Forcing Equalizer using QPSK');
grid on;
```

COMMAND WINDOW:

```
Eb/N0 = 0 dB, BER = 1.4671e-01
Eb/N0 = 2 dB, BER = 1.0812e-01
Eb/N0 = 4 dB, BER = 7.6655e-02
Eb/N0 = 6 dB, BER = 5.2740e-02
Eb/N0 = 8 dB, BER = 3.5243e-02
Eb/N0 = 10 dB, BER = 2.3075e-02
Eb/N0 = 12 dB, BER = 1.5272e-02
Eb/N0 = 14 dB, BER = 9.4300e-03
Eb/N0 = 16 dB, BER = 6.1425e-03
Eb/N0 = 18 dB, BER = 3.8875e-03
Eb/N0 = 20 dB, BER = 2.4375e-03
```

FIGURE WINDOW:



EXPERIMENT: 8

AIM: Write a MATLAB program to compute the array factor for a ULA steered towards a desired angle and prints the coherent combining SNR gain.

TOOL: MATLAB 2016

PROGRAM:

```
clc;
clear;
close all;

% ULA Parameters
N = 8;              % Number of antenna elements
d = 0.5;            % Element spacing in terms of lambda (d = lambda/2)
theta0_deg = 30;      % Desired steering angle in degrees
theta_deg = -90:0.1:90; % Scan angles in degrees

% Convert to radians
theta0 = deg2rad(theta0_deg);
theta = deg2rad(theta_deg);

% Element index
n = 0:N-1;

% Steering vector for desired angle
a0 = exp(1j * 2*pi*d*n.' * sin(theta0));   % Nx1

% Beamforming weights (conjugate steering / matched weights)
w = a0 / sqrt(N);   % Normalized weights

% Compute Array Factor
% AF(theta) = w^H * a(theta)
AF = zeros(size(theta));

for k = 1:length(theta)
    a_theta = exp(1j * 2*pi*d*n.' * sin(theta(k))); % Nx1 steering vector
    AF(k) = w' * a_theta;   % Array factor
end

% Normalize array factor
AF_mag = abs(AF);
AF_mag_norm = AF_mag / max(AF_mag);

% Convert to dB
AF_dB = 20*log10(AF_mag_norm + 1e-12); % small term avoids log(0)

% Coherent combining SNR gain
% For coherent combining of N equal-gain elements:
% SNR gain = N (linear) = 10*log10(N) dB
```

```
snr_gain_linear = N;
snr_gain_dB = 10*log10(snr_gain_linear);

fprintf('ULA with %d elements\n', N);
fprintf('Element spacing = %.2f lambda\n', d);
fprintf('Steering angle = %.2f degrees\n', theta0_deg);
fprintf('Coherent combining SNR gain = %.2f (linear)\n', snr_gain_linear);
fprintf('Coherent combining SNR gain = %.2f dB\n', snr_gain_dB);
```

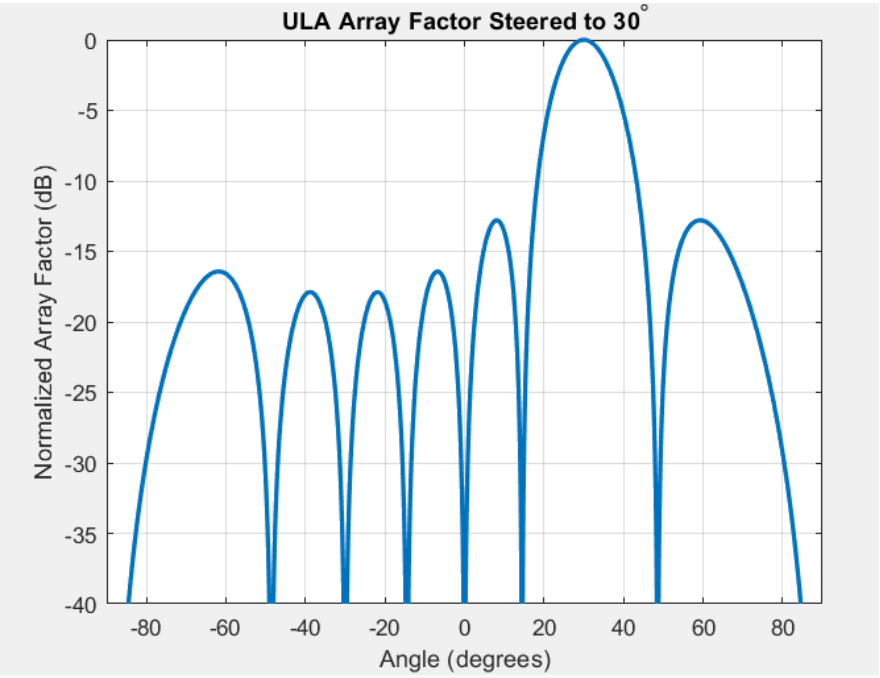
% Plot Array Factor

```
figure;
plot(theta_deg, AF_dB, 'LineWidth', 2);
grid on;
xlabel('Angle (degrees)');
ylabel('Normalized Array Factor (dB)');
title(['ULA Array Factor Steered to ', num2str(theta0_deg), '^\circ']);
axis([-90 90 -40 0]);
```

COMMAND WINDOW:

```
ULA with 8 elements
Element spacing = 0.50 lambda
Steering angle = 30.00 degrees
Coherent combining SNR gain = 8.00 (linear)
Coherent combining SNR gain = 9.03 dB
```

FIGURE WINDOW:



EXPERIMENT: 9

AIM: Write a MATLAB program to generate NR-like OFDM baseband waveform and plot its PSD.

TOOL: MATLAB 2016

PROGRAM:

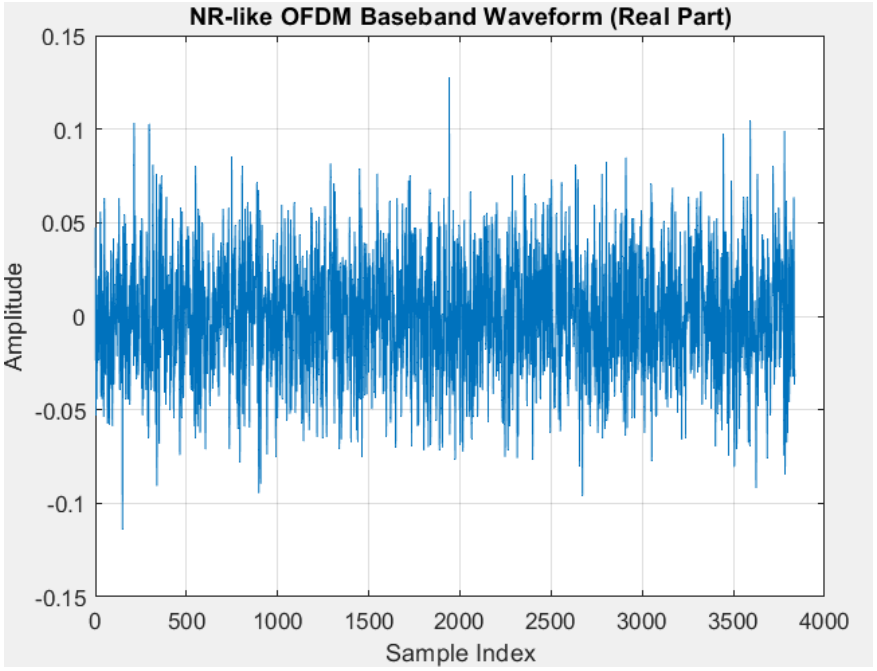
```
clc;
clear;
close all;

%% Parameters
Nfft = 256;              % FFT size
Nsc = 120;               % Number of active subcarriers
Nsym = 14;               % Number of OFDM symbols (1 slot-like)
SCS = 15e3;              % Subcarrier spacing = 15 kHz
Fs = Nfft * SCS;         % Sampling frequency
cpLen = 18;              % Cyclic prefix length
M = 4;                   % QPSK
bitsPerSym = log2(M);

%% Generate random QPSK symbols
numDataSymbols = Nsc * Nsym;
bits = randi([0 1], numDataSymbols * bitsPerSym, 1);

% QPSK mapping
bitPairs = reshape(bits, 2, []).';
qpskSymbols = (1 - 2*bitPairs(:,1)) + 1j*(1 - 2*bitPairs(:,2));
qpskSymbols = qpskSymbols / sqrt(2);   % Unit power normalization

%% Create resource grid
grid = zeros(Nfft, Nsym);

% Map active subcarriers around DC, excluding DC
half = Nsc/2;
dataMatrix = reshape(qpskSymbols, Nsc, Nsym);

% Frequency-domain subcarrier indices
% Lower half goes below DC, upper half goes above DC
grid((Nfft/2 - half + 1):(Nfft/2), :) = dataMatrix(1:half, :);
grid((Nfft/2 + 2):(Nfft/2 + half + 1), :) = dataMatrix(half+1:end, :);

%% OFDM modulation
txWaveform = [];

for symIdx = 1:Nsym
    Xk = grid(:, symIdx);

    % Shift DC to index 1 for IFFT
    Xk_ifft = ifftshift(Xk);

    % Time-domain OFDM symbol
    xn = ifft(Xk_ifft, Nfft);
```

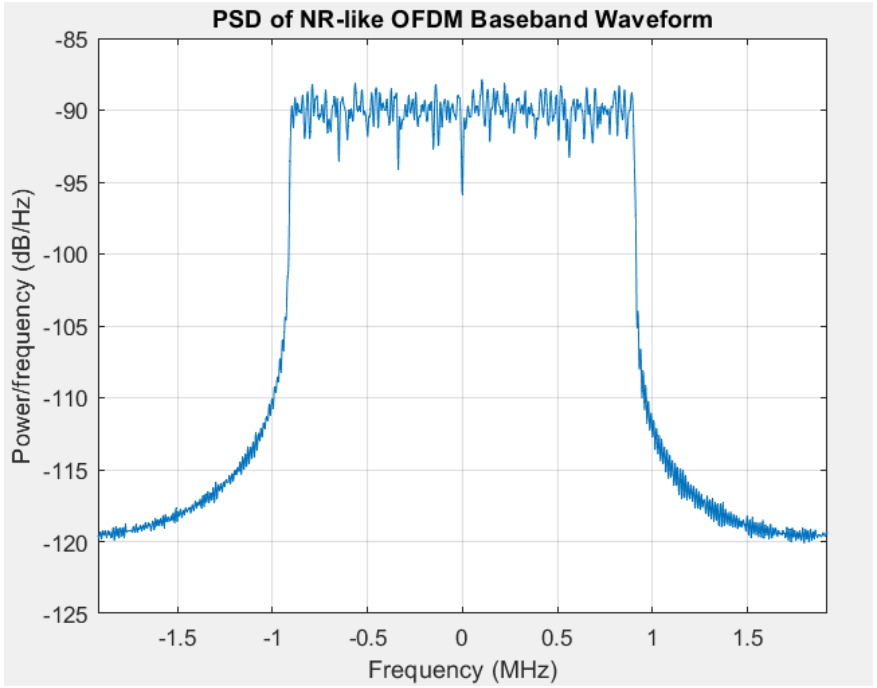
```
    % Add cyclic prefix
    xn_cp = [xn(end-cpLen+1:end); xn];

    % Concatenate symbols
    txWaveform = [txWaveform; xn_cp];
end

%% Time-domain plot
figure;
plot(real(txWaveform));
grid on;
xlabel('Sample Index');
ylabel('Amplitude');
title('NR-like OFDM Baseband Waveform (Real Part)');

%% PSD plot using Welch method
figure;
pwelch(txWaveform, hamming(512), 256, 2048, Fs, 'centered');
title('PSD of NR-like OFDM Baseband Waveform');

%% Display useful information
fprintf('FFT size             = %d\n', Nfft);
fprintf('Active subcarriers   = %d\n', Nsc);
fprintf('OFDM symbols         = %d\n', Nsym);
fprintf('Subcarrier spacing   = %.0f Hz\n', SCS);
fprintf('Sampling frequency   = %.2f MHz\n', Fs/1e6);
fprintf('Cyclic prefix length = %d samples\n', cpLen);
```

COMMAND WINDOW:

```
FFT size             = 256
Active subcarriers   = 120
OFDM symbols         = 14
Subcarrier spacing   = 15000 Hz
Sampling frequency   = 3.84 MHz
Cyclic prefix length = 18 samples
```

FIGURE WINDOW:





EXPERIMENT: 10

AIM: Write a MATLAB program to build a simple resource grid with PRB allocation and generate a waveform to display grid heatmap and spectrogram.

TOOL: MATLAB 2016

PROGRAM:

```
clc;
clear;
close all;

%% Parameters
Nfft = 512;              % FFT size
Nsym = 14;               % Number of OFDM symbols
SCS = 15e3;              % Subcarrier spacing (15 kHz)
Fs = Nfft * SCS;         % Sampling frequency
cpLen = 36;              % Cyclic prefix length
NscPerPRB = 12;          % 12 subcarriers per PRB
NPRB_total = 20;         % Total PRBs in active grid
Nsc_active = NPRB_total * NscPerPRB;
M = 4;                   % QPSK
bitsPerSym = log2(M);

%% PRB Allocation
% Example: allocate PRBs 6 to 12
allocPRB = 6:12;
numAllocPRB = length(allocPRB);

% Resource grid: active subcarriers only
% Rows = subcarriers, Columns = OFDM symbols
resourceGrid = zeros(Nsc_active, Nsym);

%% Generate QPSK symbols for allocated PRBs
numAllocatedRE = numAllocPRB * NscPerPRB * Nsym;
bits = randi([0 1], numAllocatedRE * bitsPerSym, 1);
bitPairs = reshape(bits, 2, []).';
qpskSym = ((1 - 2*bitPairs(:,1)) + 1j*(1 - 2*bitPairs(:,2))) / sqrt(2);

%% Map symbols into allocated PRBs
ptr = 1;
for prb = allocPRB
    scStart = (prb-1)*NscPerPRB + 1;
    scEnd   = prb*NscPerPRB;

    for symIdx = 1:Nsym
        resourceGrid(scStart:scEnd, symIdx) = qpskSym(ptr:ptr+NscPerPRB-1);
        ptr = ptr + NscPerPRB;
    end
end
```

```
%% Plot resource grid heatmap
figure;
imagesc(abs(resourceGrid));
axis xy;
xlabel('OFDM Symbol Index');
ylabel('Subcarrier Index');
title('Resource Grid Heatmap (Magnitude)');
colorbar;
grid on;

%% Map active resource grid into full FFT grid
% DC is kept empty
fullGrid = zeros(Nfft, Nsym);
halfAct = Nsc_active/2;

for symIdx = 1:Nsym
    % Lower half mapped below DC
    fullGrid((Nfft/2 - halfAct + 1):(Nfft/2), symIdx) = resourceGrid(1:halfAct, symIdx);

    % Upper half mapped above DC
    fullGrid((Nfft/2 + 2):(Nfft/2 + halfAct + 1), symIdx) = resourceGrid(halfAct+1:end, symIdx);
end

%% OFDM Modulation
txWaveform = [];
for symIdx = 1:Nsym
    Xk = fullGrid(:, symIdx);

    % Shift to IFFT order
    Xk_ifft = ifftshift(Xk);

    % Time domain OFDM symbol
    xn = ifft(Xk_ifft, Nfft);

    % Add cyclic prefix
    xn_cp = [xn(end-cpLen+1:end); xn];

    % Append to waveform
    txWaveform = [txWaveform; xn_cp];
end

%% Plot transmitted waveform (real part)
figure;
plot(real(txWaveform));
xlabel('Sample Index');
ylabel('Amplitude');
title('Generated OFDM Baseband Waveform (Real Part)');
grid on;

%% Spectrogram
figure;
spectrogram(txWaveform, 128, 100, 256, Fs, 'centered', 'yaxis');
title('Spectrogram of Generated OFDM Waveform');
```

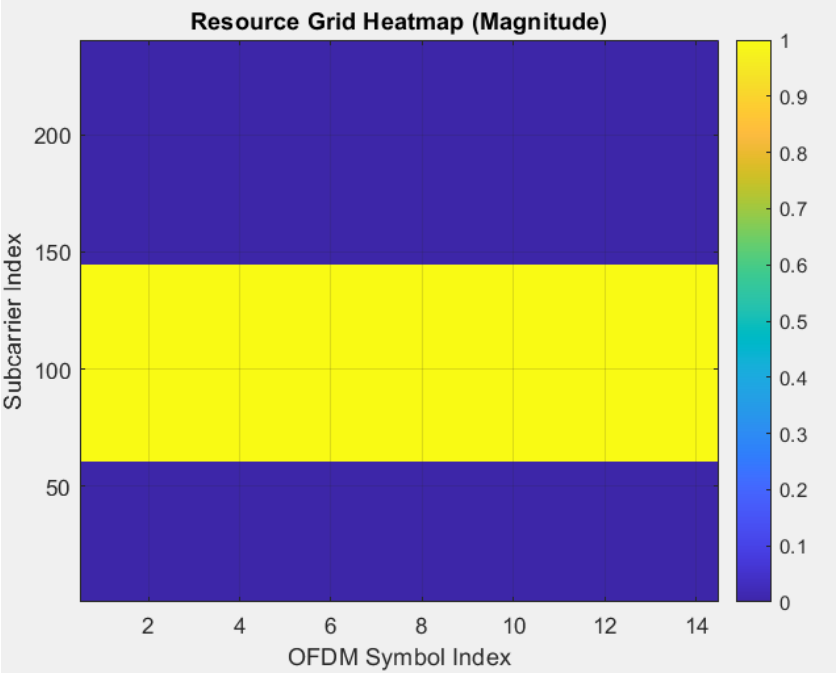
%% Display useful info

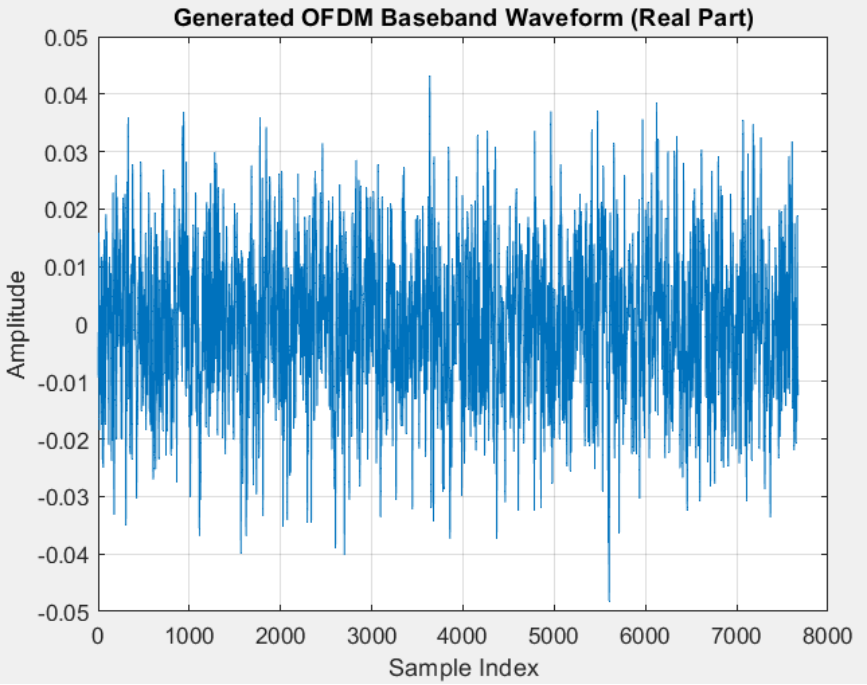
```
fprintf('FFT size                = %d\n', Nfft);
fprintf('Subcarrier spacing      = %.0f Hz\n', SCS);
fprintf('Sampling frequency      = %.2f MHz\n', Fs/1e6);
fprintf('Total PRBs in grid      = %d\n', NPRB_total);
fprintf('Allocated PRBs          = %s\n', mat2str(allocPRB));
fprintf('Active subcarriers      = %d\n', Nsc_active);
fprintf('OFDM symbols            = %d\n', Nsym);
fprintf('Waveform length (samples) = %d\n', length(txWaveform));
```

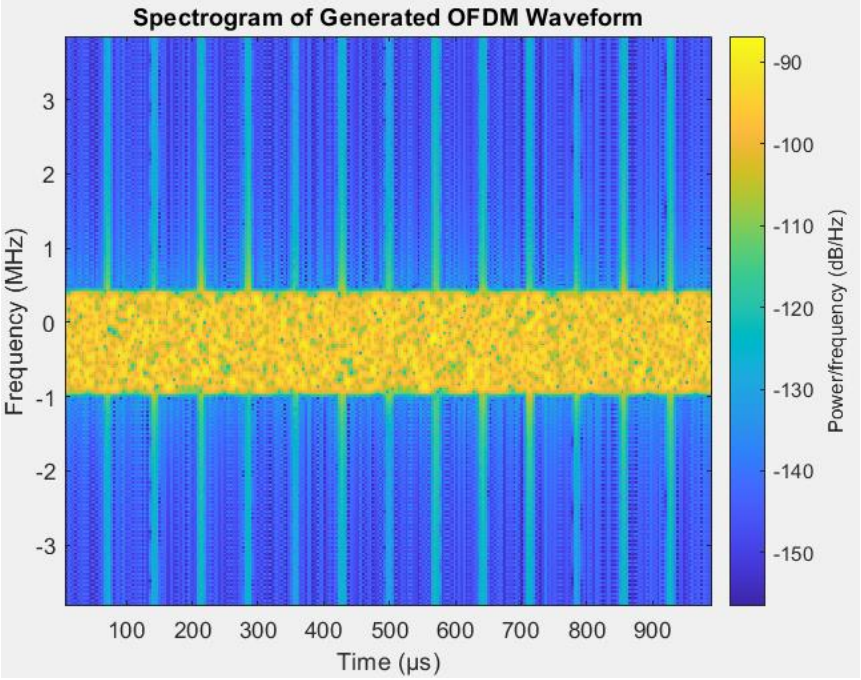
COMMAND WINDOW:

```
FFT size                = 512
Subcarrier spacing      = 15000 Hz
Sampling frequency      = 7.68 MHz
Total PRBs in grid      = 20
Allocated PRBs          = [6 7 8 9 10 11 12]
Active subcarriers      = 240
OFDM symbols            = 14
Waveform length (samples) = 7672
```

FIGURE WINDOW:




Generated OFDM Baseband Waveform (Real Part)
Amplitude
Sample Index
0.05
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
0
1000
2000
3000
4000
5000
6000
7000
8000


Spectrogram of Generated OFDM Waveform
Frequency (MHz)
Time (µs)
Power/frequency (dB/Hz)
3
2
1
0
-1
-2
-3
100
200
300
400
500
600
700
800
900
-90
-100
-110
-120
-130
-140
-150

EXPERIMENT: 11

AIM: Write a MATLAB program to plot 3GPP TR 38.901 urban microcell (Umi) LOS and NLOS path loss curves versus distance.

TOOL: MATLAB 2016

PROGRAM:

```
clc;
clear;
close all;

%% 3GPP TR 38.901 UMi Street Canyon Path Loss
% Mean path-loss curves (no random shadow fading)
% User parameters
fc_GHz = 3.5;       % Carrier frequency in GHz
hBS = 10;           % BS height in meters (UMi default)
hUT = 1.5;          % UT height in meters
hE  = 1.0;          % Effective environment height for UMi
c   = 3e8;          % Speed of light (m/s)

% Distance range
d2D = linspace(10, 5000, 2000);   % 2D distance in meters
d3D = sqrt(d2D.^2 + (hBS - hUT).^2);

% Effective breakpoint distance
% dBP' = 4*hBS'*hUT'*fc/c
hBS_eff = hBS - hE;
hUT_eff = hUT - hE;
fc_Hz = fc_GHz * 1e9;
dBP = 4 * hBS_eff * hUT_eff * fc_Hz / c;

% UMi LOS path loss
PL1 = 32.4 + 21*log10(d3D) + 20*log10(fc_GHz);

PL2 = 32.4 + 40*log10(d3D) + 20*log10(fc_GHz) ...
- 9.5*log10(dBP^2 + (hBS - hUT)^2);

PL_LOS = PL1;
idxBP = d2D > dBP;
PL_LOS(idxBP) = PL2(idxBP);

% UMi NLOS path loss
PL_NLOS_prime = 35.3*log10(d3D) + 22.4 + 21.3*log10(fc_GHz) ...
            - 0.3*(hUT - 1.5);

PL_NLOS = max(PL_LOS, PL_NLOS_prime);
```

% Plot

```
figure;
semilogx(d2D, PL_LOS, 'b', 'LineWidth', 2); hold on;
semilogx(d2D, PL_NLOS, 'r', 'LineWidth', 2);

yl = ylim;
semilogx([dBP dBP], yl, 'k--', 'LineWidth', 1.5);
ylim(yl);

grid on;
xlabel('Distance d_{2D} (m)');
ylabel('Path Loss (dB)');
title(['3GPP TR 38.901 UMi Path Loss, f_c = ' num2str(fc_GHz) ' GHz']);
legend('UMi LOS', 'UMi NLOS', 'Breakpoint d"_{BP}', 'Location', 'NorthWest');
```

% Print values

```
fprintf('3GPP TR 38.901 UMi Street Canyon Path Loss Model\n');
fprintf('Carrier frequency      = %.2f GHz\n', fc_GHz);
fprintf('BS height              = %.2f m\n', hBS);
fprintf('UT height              = %.2f m\n', hUT);
fprintf('Effective height hE    = %.2f m\n', hE);
fprintf('Breakpoint distance    = %.2f m\n', dBP);
```

% Example values at selected distances

```
testD = [10 50 100 200 500 1000];
testD3D = sqrt(testD.^2 + (hBS - hUT).^2);

PL1_t = 32.4 + 21*log10(testD3D) + 20*log10(fc_GHz);
PL2_t = 32.4 + 40*log10(testD3D) + 20*log10(fc_GHz) ...
       - 9.5*log10(dBP^2 + (hBS - hUT)^2);

PL_LOS_t = PL1_t;
PL_LOS_t(testD > dBP) = PL2_t(testD > dBP);

PL_NLOS_prime_t = 35.3*log10(testD3D) + 22.4 + 21.3*log10(fc_GHz) ...
                - 0.3*(hUT - 1.5);

PL_NLOS_t = max(PL_LOS_t, PL_NLOS_prime_t);

fprintf('\nDistance(m)    LOS(dB)    NLOS(dB)\n');
fprintf('-----------------------------------\n');
for k = 1:length(testD)
   fprintf('%8.0f    %8.2f    %8.2f\n', testD(k), PL_LOS_t(k), PL_NLOS_t(k));
end
```

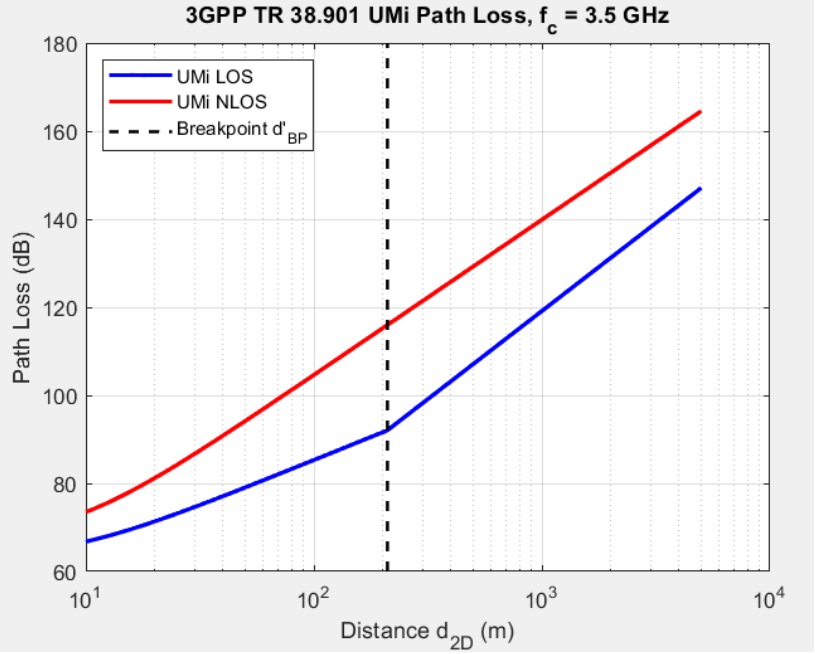
COMMAND WINDOW:

```
  3GPP TR 38.901 UMi Street Canyon Path Loss Model
  Carrier frequency      = 3.50 GHz
  BS height              = 10.00 m
  UT height              = 1.50 m
```

Effective height hE = 1.00 m
Breakpoint distance = 210.00 m

Distance(m)	LOS(dB)	NLOS(dB)
10	66.76	73.46
50	79.09	94.18
100	85.31	104.64
200	91.61	115.23
500	107.11	129.26
1000	119.15	139.89

FIGURE WINDOW:



EXPERIMENT: 12

AIM: Write a MATLAB program for: 5G URLLC for Industrial Automation.

TOOL: MATLAB 2016

PROGRAM:

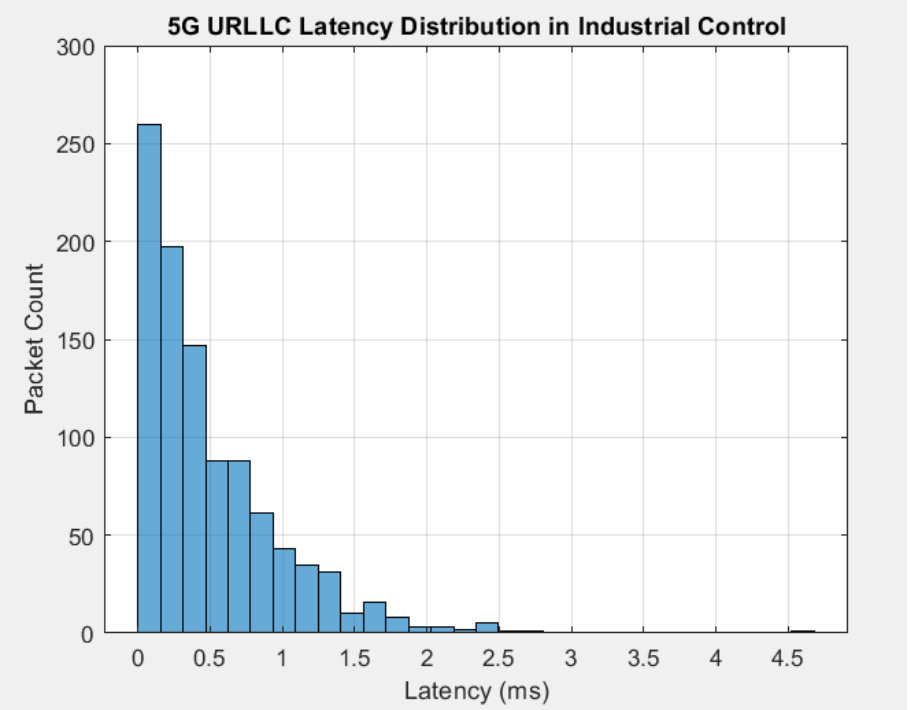
```
clc;
clear;
close all;

% Parameters
numPackets = 1000;            % Number of control messages
packetSizeBytes = 32;         % Packet size in bytes
targetLatency_ms = 1;         % Target 5G URLLC latency in ms
targetReliability = 1 - 1e-5; % Target reliability = 99.999%

% Simulated latency and error
meanLatency = 0.5;            % Average latency in ms
packetLossRate = 1e-4;        % Simulated packet loss probability

% Simulation
success = 0;
latencies = zeros(numPackets,1);

for i = 1:numPackets
    latency = exprnd(meanLatency);      % Exponential latency variation
    successFlag = rand > packetLossRate;

    if latency <= targetLatency_ms && successFlag
        success = success + 1;
    end
    latencies(i) = latency;
end

% Results
reliability = success / numPackets;
fprintf('Packet Size: %d bytes\n', packetSizeBytes);
fprintf('Packet Success Rate (Reliability): %.6f\n', reliability);
fprintf('Average Latency: %.3f ms\n', mean(latencies));

if reliability >= targetReliability
    fprintf('URLLC reliability target achieved.\n');
else
    fprintf('URLLC reliability target NOT achieved.\n');
end

% Plot
figure;
histogram(latencies, 30);
xlabel('Latency (ms)');
```

```
ylabel('Packet Count');
title('5G URLLC Latency Distribution in Industrial Control');
grid on;
```

COMMAND WINDOW:

```
Packet Size: 32 bytes
Packet Success Rate (Reliability): 0.872000
Average Latency: 0.499 ms
URLLC reliability target NOT achieved.
```

FIGURE WINDOW:



EXPERIMENT: 13

AIM: Write a MATLAB program that simulates network slicing in 5G, where different types of services (e.g., eMBB, URLLC, mMTC) share the same physical infrastructure but are logically separated based on their QoS requirements.

TOOL: MATLAB 2016

PROGRAM:

```
clc;
clear;
close all;

% Parameters
totalBandwidth = 100;   % MHz (total available spectrum)

% Slice definitions (percentages of total BW)
slices = struct( ...
    'eMBB', struct('share', 0.5, 'latencyReq', 10, 'throughputReq', 100), ...
    'URLLC', struct('share', 0.2, 'latencyReq', 1,  'throughputReq', 10), ...
    'mMTC', struct('share', 0.3, 'latencyReq', 50, 'throughputReq', 1) ...
);

% Allocate bandwidth
slices.eMBB.bw  = slices.eMBB.share  * totalBandwidth;
slices.URLLC.bw = slices.URLLC.share * totalBandwidth;
slices.mMTC.bw  = slices.mMTC.share  * totalBandwidth;

% Time axis
time = 1:100;

% Simulate traffic arrival (random for each slice)
traffic.eMBB  = 80  + 20*randn(1,100);   % Mbps
traffic.URLLC = 8   + 2*randn(1,100);    % Mbps
traffic.mMTC  = 0.8 + 0.5*randn(1,100);  % Mbps

% Clip negative traffic
traffic.eMBB(traffic.eMBB < 0)   = 0;
traffic.URLLC(traffic.URLLC < 0) = 0;
traffic.mMTC(traffic.mMTC < 0)   = 0;

% Display allocated bandwidth
fprintf('Allocated Bandwidth:\n');
fprintf('eMBB  = %.2f MHz\n', slices.eMBB.bw);
fprintf('URLLC = %.2f MHz\n', slices.URLLC.bw);
fprintf('mMTC  = %.2f MHz\n', slices.mMTC.bw);

% Plot traffic and throughput requirement
figure;
```

```
subplot(3,1,1);
plot(time, traffic.eMBB, 'b', 'LineWidth', 1.5); hold on;
plot(time, slices.eMBB.throughputReq * ones(size(time)), '--r', 'LineWidth', 1.5);
title('eMBB Slice Traffic');
ylabel('Mbps');
legend('Traffic', 'Throughput Requirement');
grid on;

subplot(3,1,2);
plot(time, traffic.URLLC, 'b', 'LineWidth', 1.5); hold on;
plot(time, slices.URLLC.throughputReq * ones(size(time)), '--r', 'LineWidth', 1.5);
title('URLLC Slice Traffic');
ylabel('Mbps');
legend('Traffic', 'Throughput Requirement');
grid on;

subplot(3,1,3);
plot(time, traffic.mMTC, 'b', 'LineWidth', 1.5); hold on;
plot(time, slices.mMTC.throughputReq * ones(size(time)), '--r', 'LineWidth', 1.5);
title('mMTC Slice Traffic');
xlabel('Time');
ylabel('Mbps');
legend('Traffic', 'Throughput Requirement');
grid on;
```

COMMAND WINDOW:

```
Allocated Bandwidth:
eMBB  = 50.00 MHz
URLLC = 20.00 MHz
mMTC  = 30.00 MHz
```

FIGURE WINDOW:

EXPERIMENT: 14

AIM: Write a MATLAB program for RB scheduler with URLLC pre-emption and proportional sharing for eMBB/mMTC to simulate arrival and backlog over time.

TOOL: MATLAB 2016

PROGRAM:

```
clc;
clear;
close all;

%% Parameters
T = 200;              % Number of time slots
totalRB = 50;         % Total RBs available per slot

% Optional proportional weights for non-URLLC traffic
w_eMBB = 3;           % Higher share preference for eMBB
w_mMTC = 1;           % Lower share preference for mMTC

%% Pre-allocation for arrays
arrURLLC = zeros(T,1);
arreMBB  = zeros(T,1);
arrmMTC  = zeros(T,1);

allocURLLC = zeros(T,1);
alloceMBB  = zeros(T,1);
allocmMTC  = zeros(T,1);

backlogURLLC = zeros(T,1);
backlogeMBB  = zeros(T,1);
backlogmMTC  = zeros(T,1);

%% Initial queue values
qURLLC = 0;
qeMBB  = 0;
qmMTC  = 0;

%% Simulation over time
for t = 1:T
    % Traffic arrivals per slot
    % URLLC: usually small, occasional bursts
    arrURLLC(t) = randi([0 4]);
```

```
if rand < 0.08
   arrURLLC(t) = arrURLLC(t) + randi([5 12]);
end

% eMBB: higher and smoother load
arreMBB(t) = randi([12 25]);

% mMTC: low-to-moderate random arrivals
arrmMTC(t) = randi([3 10]);

% Add arrivals to queues
qURLLC = qURLLC + arrURLLC(t);
qeMBB  = qeMBB  + arreMBB(t);
qmMTC  = qmMTC  + arrmMTC(t);

% Step 1: URLLC pre-emption
allocURLLC(t) = min(qURLLC, totalRB);
qURLLC = qURLLC - allocURLLC(t);
remainingRB = totalRB - allocURLLC(t);

% Step 2: Proportional sharing
% between eMBB and mMTC
if remainingRB > 0
   % Weighted backlog-based proportional share
   demandMetric_eMBB = w_eMBB * qeMBB;
   demandMetric_mMTC = w_mMTC * qmMTC;
   totalMetric = demandMetric_eMBB + demandMetric_mMTC;
   if totalMetric > 0
      share_eMBB = remainingRB * (demandMetric_eMBB / totalMetric);
      share_mMTC = remainingRB * (demandMetric_mMTC / totalMetric);
   else
      share_eMBB = 0;
      share_mMTC = 0;
   end

   % Integer RB allocation
   alloceMBB(t) = min(qeMBB, floor(share_eMBB));
   allocmMTC(t) = min(qmMTC, floor(share_mMTC));

   % Remaining RBs after floor() are distributed one-by-one
   usedRB = alloceMBB(t) + allocmMTC(t);
   leftoverRB = remainingRB - usedRB;
```

```
        while leftoverRB > 0
        % Give leftover RBs to the slice with larger unmet weighted need
        unmet_eMBB = max(qeMBB - alloceMBB(t), 0) * w_eMBB;
        unmet_mMTC = max(qmMTC - allocmMTC(t), 0) * w_mMTC;
        if unmet_eMBB == 0 && unmet_mMTC == 0
            break;
        end
        if unmet_eMBB >= unmet_mMTC && alloceMBB(t) < qeMBB
            alloceMBB(t) = alloceMBB(t) + 1;
        elseif allocmMTC(t) < qmMTC
            allocmMTC(t) = allocmMTC(t) + 1;
        elseif alloceMBB(t) < qeMBB
            alloceMBB(t) = alloceMBB(t) + 1;
        else
            break;
        end
        leftoverRB = leftoverRB - 1;
      end

      % Update queues
      qeMBB = qeMBB - alloceMBB(t);
      qmMTC = qmMTC - allocmMTC(t);
   end

   % Store backlog after service
   backlogURLLC(t) = qURLLC;
   backlogeMBB(t)  = qeMBB;
   backlogmMTC(t)  = qmMTC;
end

%% Display summary
fprintf('Simulation finished for %d time slots\n', T);
fprintf('Total RBs per slot = %d\n\n', totalRB);

fprintf('Average arrival per slot:\n');
fprintf('URLLC = %.2f RB-units\n', mean(arrURLLC));
fprintf('eMBB  = %.2f RB-units\n', mean(arreMBB));
fprintf('mMTC  = %.2f RB-units\n\n', mean(arrmMTC));

fprintf('Average allocation per slot:\n');
fprintf('URLLC = %.2f RBs\n', mean(allocURLLC));
fprintf('eMBB  = %.2f RBs\n', mean(alloceMBB));
fprintf('mMTC  = %.2f RBs\n\n', mean(allocmMTC));
```

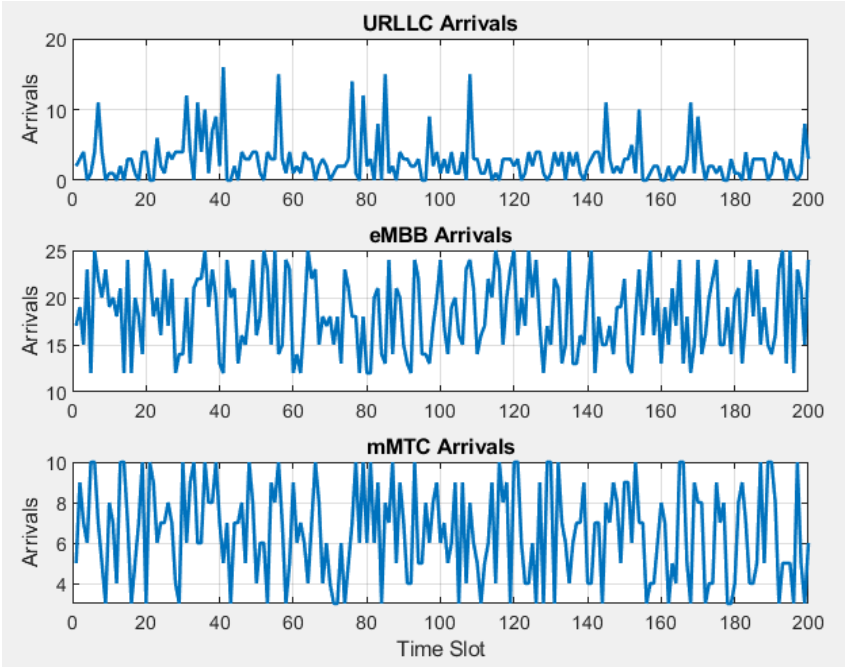
```
fprintf('Final backlog:\n');
fprintf('URLLC = %d\n', backlogURLLC(end));
fprintf('eMBB  = %d\n', backlogeMBB(end));
fprintf('mMTC  = %d\n', backlogmMTC(end));

%% Plot arrivals
figure;
subplot(3,1,1);
plot(1:T, arrURLLC, 'LineWidth', 1.5);
ylabel('Arrivals');
title('URLLC Arrivals');
grid on;

subplot(3,1,2);
plot(1:T, arreMBB, 'LineWidth', 1.5);
ylabel('Arrivals');
title('eMBB Arrivals');
grid on;

subplot(3,1,3);
plot(1:T, arrmMTC, 'LineWidth', 1.5);
xlabel('Time Slot');
ylabel('Arrivals');
title('mMTC Arrivals');
grid on;

%% Plot RB allocations
figure;
subplot(3,1,1);
stairs(1:T, allocURLLC, 'LineWidth', 1.5);
ylabel('RBs');
title('URLLC Allocation (Pre-emptive)');
grid on;

subplot(3,1,2);
stairs(1:T, alloceMBB, 'LineWidth', 1.5);
ylabel('RBs');
title('eMBB Allocation');
grid on;

subplot(3,1,3);
stairs(1:T, allocmMTC, 'LineWidth', 1.5);
xlabel('Time Slot');
```

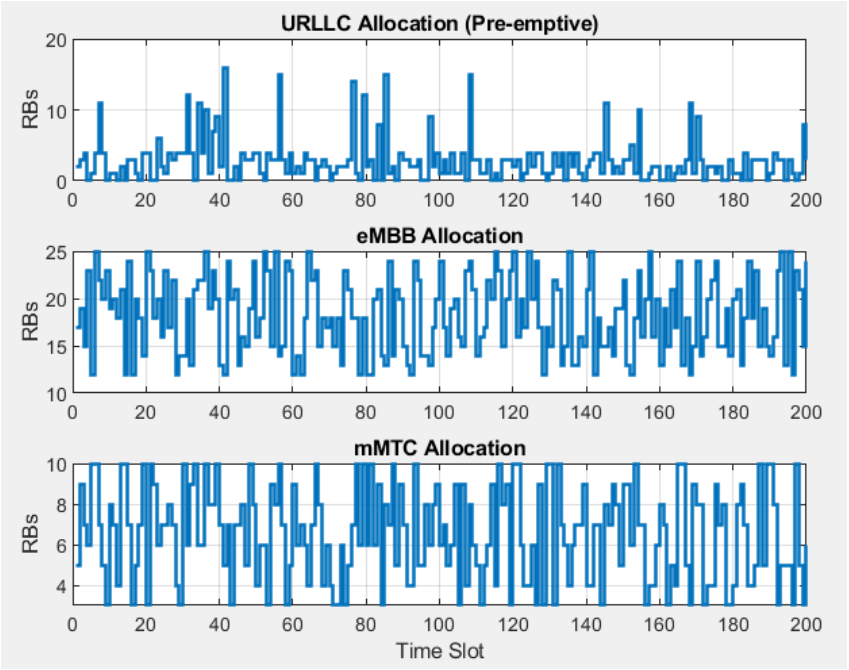
```
ylabel('RBs');
title('mMTC Allocation');
grid on;

%% Plot backlog evolution
figure;
plot(1:T, backlogURLLC, 'LineWidth', 1.8); hold on;
plot(1:T, backlogeMBB,  'LineWidth', 1.8);
plot(1:T, backlogmMTC,  'LineWidth', 1.8);
xlabel('Time Slot');
ylabel('Backlog (RB-units)');
title('Backlog Evolution Over Time');
legend('URLLC', 'eMBB', 'mMTC');
grid on;

%% Combined stacked allocation view
figure;
bar(1:T, [allocURLLC alloceMBB allocmMTC], 'stacked');
xlabel('Time Slot');
ylabel('Allocated RBs');
title('Stacked RB Allocation per Slot');
legend('URLLC', 'eMBB', 'mMTC');
grid on;
```

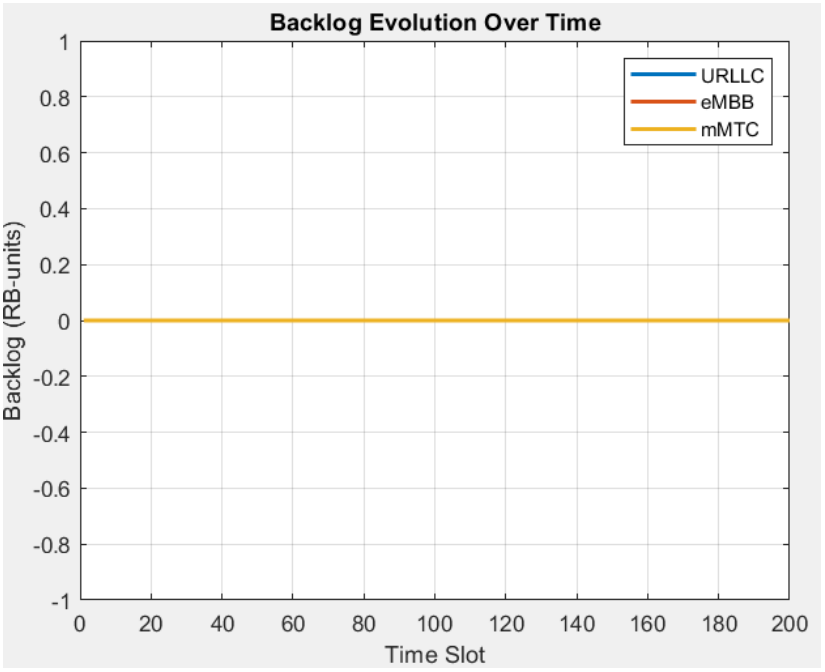
COMMAND WINDOW:

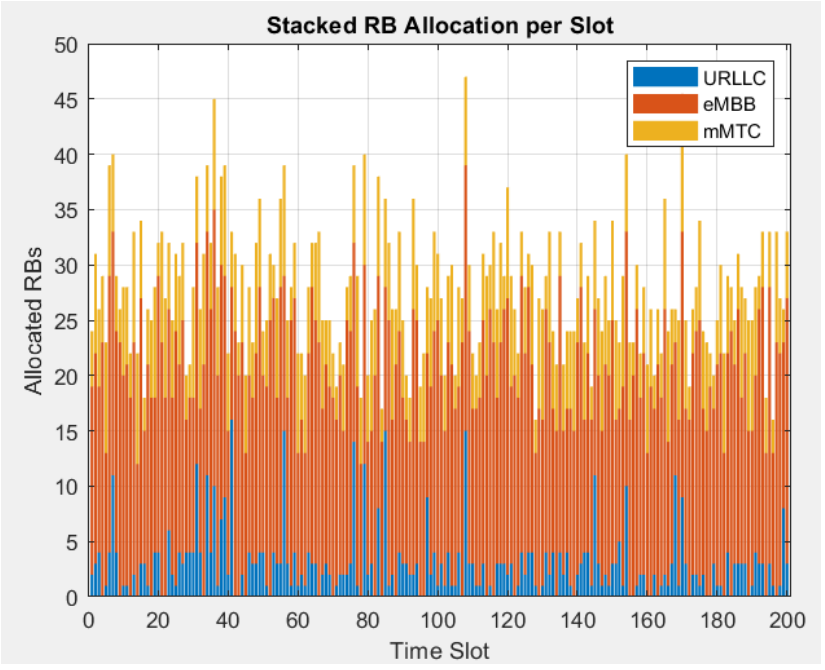
```
Simulation finished for 200 time slots
Total RBs per slot = 50
Average arrival per slot:
URLLC = 2.84 RB-units
eMBB  = 18.30 RB-units
mMTC  = 6.54 RB-units
Average allocation per slot:
URLLC = 2.84 RBs
eMBB  = 18.30 RBs
mMTC  = 6.54 RBs
Final backlog:
URLLC = 0
eMBB  = 0
mMTC  = 0
```

FIGURE WINDOW:






Backlog Evolution Over Time
URLLC
eMBB
mMTC
Backlog (RB-units)
Time Slot
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120
140
160
180
200


Stacked RB Allocation per Slot
URLLC
eMBB
mMTC
Allocated RBs
Time Slot
50
45
40
35
30
25
20
15
10
5
0
0
20
40
60
80
100
120
140
160
180
200

EXPERIMENT: 15

AIM: Write a MATLAB program for simple IOT data transmission.

TOOL: MATLAB 2016

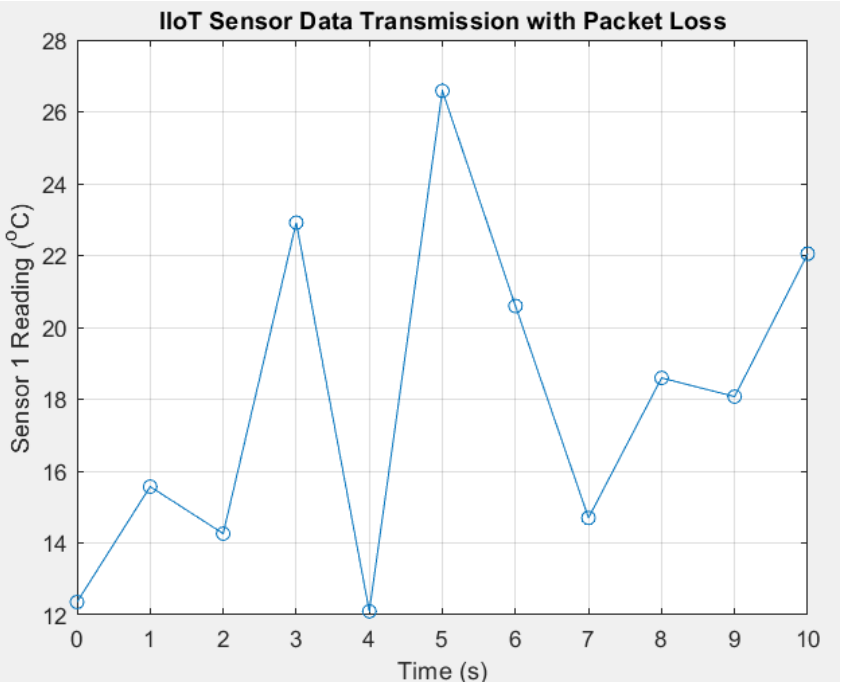
PROGRAM:

```
clc;
clear;
close all;

% Parameters
numSensors = 5;             % Number of IIoT sensors
samplingRate = 1;           % Hz
duration = 10;              % seconds
time = 0:1/samplingRate:duration;
dataSizePerSample = 100;    % Bytes per reading

% Simulate sensor readings (e.g., temperature)
sensorData = 20 + 5 * randn(length(time), numSensors);

% Simulate transmission to controller (with 1% packet loss)
packetLossRate = 0.01;
receivedData = sensorData;

lossMatrix = rand(size(sensorData)) < packetLossRate;
receivedData(lossMatrix) = NaN;    % Mark lost packets as NaN

% Display summary
totalPackets = numel(sensorData);
lostPackets = sum(lossMatrix(:));
receivedPackets = totalPackets - lostPackets;

fprintf('Number of sensors       = %d\n', numSensors);
fprintf('Sampling rate           = %d Hz\n', samplingRate);
fprintf('Duration                = %d s\n', duration);
fprintf('Total transmitted packets= %d\n', totalPackets);
fprintf('Lost packets            = %d\n', lostPackets);
fprintf('Packet delivery ratio   = %.2f%%\n', 100*receivedPackets/totalPackets);

% Plot data from one sensor
figure;
plot(time, receivedData(:,1), '-o');
xlabel('Time (s)');
ylabel('Sensor 1 Reading (^oC)');
title('IIoT Sensor Data Transmission with Packet Loss');
grid on;
```

COMMAND WINDOW:

Number of sensors = 5
Sampling rate = 1 Hz
Duration = 10 s
Total transmitted packets= 55
Lost packets = 0
Packet delivery ratio = 100.00%

FIGURE WINDOW:



EXPERIMENT: 16

AIM: Write a MATLAB program to estimate user throughput for Carrier aggregation with two components carriers (CA_2CC).

TOOL: MATLAB 2016

PROGRAM:

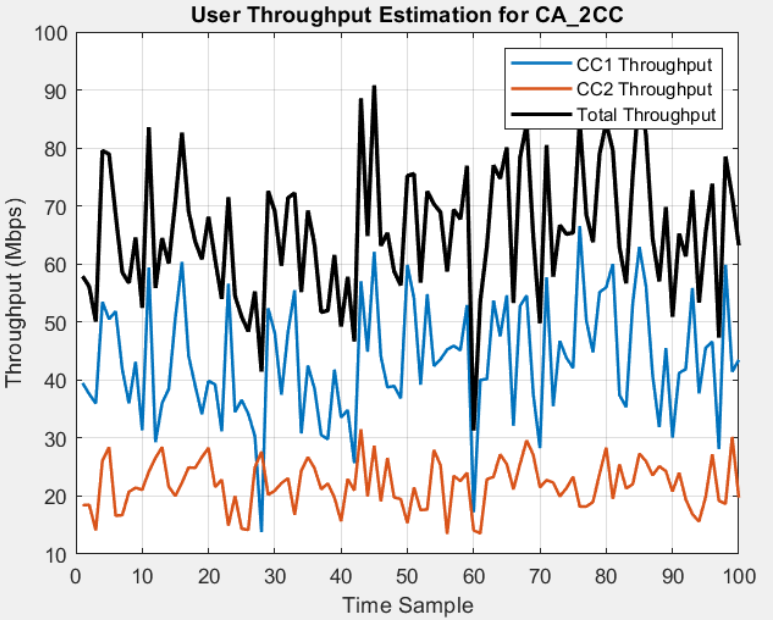
```
clc;
clear;
close all;

%% Parameters
numUsers = 1;              % Single user throughput estimation
numSamples = 100;          % Time samples

% Component Carrier bandwidths
BW1_MHz = 20;              % CC1 bandwidth in MHz
BW2_MHz = 15;              % CC2 bandwidth in MHz

BW1 = BW1_MHz * 1e6;       % Convert to Hz
BW2 = BW2_MHz * 1e6;

% Overhead factor (control, pilots, guard, etc.)
overhead = 0.20;           % 20% overhead

% Scheduler share for the user on each carrier
share1 = 0.8;              % 80% of CC1 resources
share2 = 0.7;              % 70% of CC2 resources

%% Simulated SINR values over time (in dB)
SINR1_dB = 10 + 3*randn(numSamples,1);   % CC1 SINR
SINR2_dB = 7  + 2*randn(numSamples,1);   % CC2 SINR

% Limit SINR to reasonable range
SINR1_dB(SINR1_dB < -5) = -5;
SINR2_dB(SINR2_dB < -5) = -5;

%% Convert SINR from dB to linear
SINR1 = 10.^(SINR1_dB/10);
SINR2 = 10.^(SINR2_dB/10);

%% Spectral efficiency estimation using Shannon-like formula
% eta = log2(1 + SINR)
eta1 = log2(1 + SINR1);      % bits/s/Hz
eta2 = log2(1 + SINR2);

%% Throughput per component carrier
throughput1_bps = BW1 .* eta1 * share1 * (1 - overhead);
throughput2_bps = BW2 .* eta2 * share2 * (1 - overhead);
```

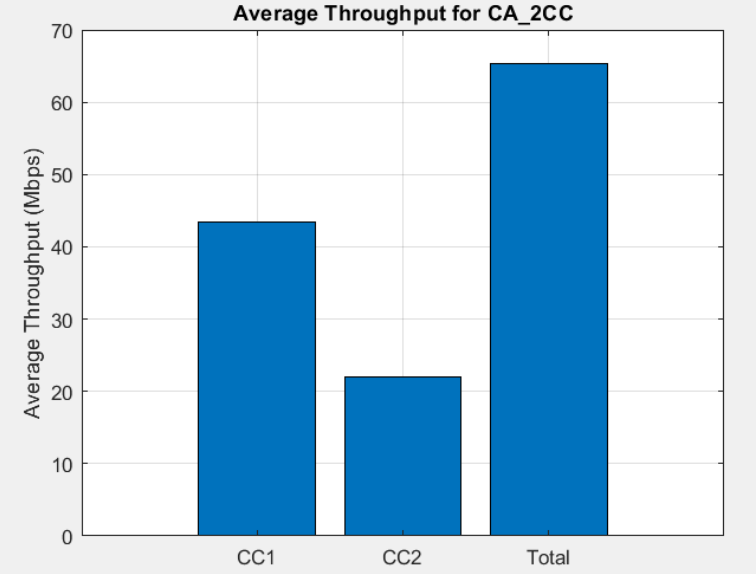
```
%% Total CA throughput
totalThroughput_bps = throughput1_bps + throughput2_bps;

%% Convert to Mbps
throughput1_Mbps = throughput1_bps / 1e6;
throughput2_Mbps = throughput2_bps / 1e6;
totalThroughput_Mbps = totalThroughput_bps / 1e6;

%% Display average results
fprintf('Carrier Aggregation with 2 Component Carriers (CA_2CC)\n');
fprintf('-----------------------------------------------------\n');
fprintf('CC1 Bandwidth = %.1f MHz\n', BW1_MHz);
fprintf('CC2 Bandwidth = %.1f MHz\n', BW2_MHz);
fprintf('Average CC1 Throughput = %.2f Mbps\n', mean(throughput1_Mbps));
fprintf('Average CC2 Throughput = %.2f Mbps\n', mean(throughput2_Mbps));
fprintf('Average Total Throughput = %.2f Mbps\n', mean(totalThroughput_Mbps));

%% Plot throughput over time
time = 1:numSamples;

figure;
plot(time, throughput1_Mbps, 'LineWidth', 1.5); hold on;
plot(time, throughput2_Mbps, 'LineWidth', 1.5);
plot(time, totalThroughput_Mbps, 'k', 'LineWidth', 2);
grid on;
xlabel('Time Sample');
ylabel('Throughput (Mbps)');
title('User Throughput Estimation for CA\_2CC');
legend('CC1 Throughput', 'CC2 Throughput', 'Total Throughput');

%% Bar chart of average throughput
figure;
bar([mean(throughput1_Mbps), mean(throughput2_Mbps), mean(totalThroughput_Mbps)]);
set(gca, 'XTickLabel', {'CC1','CC2','Total'});
ylabel('Average Throughput (Mbps)');
title('Average Throughput for CA\_2CC');
grid on;
```

COMMAND WINDOW:

```
Carrier Aggregation with 2 Component Carriers (CA_2CC)
-----------------------------------------------------
CC1 Bandwidth = 20.0 MHz
CC2 Bandwidth = 15.0 MHz
Average CC1 Throughput = 43.37 Mbps
Average CC2 Throughput = 22.02 Mbps
Average Total Throughput = 65.38 Mbps
```

FIGURE WINDOW:





EXPERIMENT: 17

AIM: Demonstrate the creation of pcap file, reading of pcap file and packet truncation with variable snap length.

H/W Requirement: Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz, 16 CPU, 32 GB RAM

S/W Requirement: Ubuntu 18.04 LTS, Docker latest version, docker-compose (1.29.2)

CODE:

"tcpdump" => used for capturing the incoming & outgoing traffic
at network interface level
(network traffic)

tcpdump -i any -s0 -w /tmp/test.pcap

tcpdump -i any -s0
ping 127.0.0.1

tcpdump -i <interface_name> -s0
↓
interface name

ping 192.168.56.101 -> (generate some traffic)

tcpdump -i <interface_name> -s0 -w test.pcap
↓
interface name

EXPERIMENT: 18

AIM: Demonstrate the packet capturing and analysis using WIRESHARK tool by filtering the TCP and UDP packets.

H/W Requirement: Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz, 16 CPU, 32 GB RAM

S/W Requirement: Ubuntu 18.04 LTS, Docker latest version, docker-compose (1.29.2)

CODE:

"tcpdump" => used for capturing the incoming & outgoing traffic
at network interface level
(network traffic)

tcpdump -i any -s0 -w /tmp/test.pcap

tcpdump -i any -s0
ping 127.0.0.1

tcpdump -i <interface_name> -s0
↓
interface name

ping 192.168.56.101 -> (generate some traffic)

tcpdump -i <interface_name> -s0 -w test.pcap
↓
interface name

sudo wireshark -

nmap -vv -p 80 hackthissite.org

* wireshark -i eno5 -k

sudo apt-get install wireshark
sudo apt update
sudo apt upgrade
wireshark -i eno5 -k

APPENDIX

Demonstrate OAI Full Stack 5G-NR RF simulation with containers.

OAI LINK : ci-scripts/yaml_files/5g_rfsimulator/README.md · develop · oai / openairinterface5G · GitLab (eurecom.fr)

H/W Requirement : Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz , 16 CPU, 32 GB RAM

S/W Requirement: Ubuntu 18.04 LTS, Docker latest version, docker-compose (1.29.2

OAI Full Stack 5G-NR RF simulation with containers

OAI LINK : ci-scripts/yaml_files/5g_rfsimulator/README.md · develop · oai / openairinterface5G · GitLab (eurecom.fr)

H/W Requirement : Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz , 16 CPU, 32 GB RAM

S/W Requirement: Ubuntu 18.04 LTS, Docker latest version, docker-compose (1.29.2)

,

1. How to start

Clone Open Air Interface repository and check out develop branch. (Only develop branch is controlled for 5G SA deployment).

git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git

git checkout develop

Once we have check out develop branch, go to below mentioned directory.

openairinterface5g/ci-scripts/yaml_files/5g_f1_rfsimulator

We will fine 5G SA based deployment file (docker-compose.yaml) along with mysql configuration files.

```
root@ubuntu:/home/vikash/openairinterface5g/ci-scripts/yaml_files/5g_f1_rfsimulator# ls
amf-healthcheck.sh    mysql-healthcheck.sh  oai_db.sql          spgwu-healthcheck.sh
docker-compose.yaml   nrf-healthcheck.sh    smf-healthcheck.sh
```

Run below command to pull images from docker repo and tag those images.

docker pull mysql:5.7

docker pull rdefosseoai/oai-amf:latest

docker pull rdefosseoai/oai-nrf:latest

docker pull rdefosseoai/oai-smf:latest

docker pull rdefosseoai/oai-spgwu-tiny:latest

docker pull rdefosseoai/oai-gnb:develop

docker pull rdefosseoai/oai-nr-ue:develop

If you facing issue while pulling the docker images, please try “docker login” with correct credentials.

Once we pulled docker images, run below commands to tag images

docker image tag rdefosseoai/oai-amf:latest oai-amf:latest

docker image tag rdefosseoai/oai-nrf:latest oai-nrf:latest

docker image tag rdefosseoai/oai-smf:latest oai-smf:latest

docker image tag rdefosseoai/oai-spgwu-tiny:latest oai-spgwu-tiny:latest

docker image tag rdefosseoai/oai-gnb:develop oai-gnb:develop

docker image tag rdefosseoai/oai-nr-ue:develop oai-nr-ue:develop

2. Deploy OAI 5G Core Network

```
$ cd ci-scripts/yaml_files/5g_f1_rfsimulator
$ docker-compose up -d mysql oai-nrf oai-amf oai-smf oai-spgwu oai-ext-dn
Creating network "rfsim5g-oai-public-net" with driver "bridge"
Creating network "rfsim5g-oai-traffic_net-net" with driver "bridge"
Creating rfsim5g-oai-nrf ... done
Creating rfsim5g-mysql      ... done
Creating rfsim5g-oai-spgwu ... done
Creating rfsim5g-oai-amf   ... done
Creating rfsim5g-oai-smf   ... done
Creating rfsim5g-oai-ext-dn ... done
```

Wait for a bit.

```
$ docker-compose ps -a
       Name                     Command                  State
Ports
-------------------------------------------------------------------------------------------------------
-------------------------------
rfsim5g-mysql      docker-entrypoint.sh  mysqld     Up  (healthy)
3306/tcp, 33060/tcp
rfsim5g-oai-amf    /bin/bash  /openair-amf/bin  ...  Up  (healthy)
38412/sctp, 80/tcp, 9090/tcp
rfsim5g-oai-ext-dn  /bin/bash  -c  apt  update;  ...  Up  (healthy)
rfsim5g-oai-nrf    /bin/bash  /openair-nrf/bin  ...  Up  (healthy)
80/tcp, 9090/tcp
rfsim5g-oai-smf    /bin/bash  -c  /openair-smf/  ...  Up  (healthy)
80/tcp, 8805/udp, 9090/tcp
rfsim5g-oai-spgwu   /openair-spgwu-tiny/bin/en  ...  Up  (healthy)
2152/udp, 8805/udp
```

At this point, you can prepare a capture on the newly-created public docker bridges:

```
$ ifconfig
...
rfsim5g-public: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.71.129  netmask 255.255.255.192  broadcast
192.168.71.191
        inet6 fe80::42:c4ff:fe2b:3d38  prefixlen 64  scopeid 0x20<link>
        ether 02:42:c4:2b:3d:38  txqueuelen 0  (Ethernet)
        RX packets 4  bytes 112 (112.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7  bytes 626 (626.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

rfsim5g-traffic: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.72.129  netmask 255.255.255.192  broadcast
192.168.72.191
        inet6 fe80::42:b5ff:fed3:e732  prefixlen 64  scopeid 0x20<link>
```

```
        ether 02:42:b5:d3:e7:32  txqueuelen 0  (Ethernet)
        RX packets 2652  bytes 142335 (142.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3999  bytes 23367972 (23.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
...
```

3. Deploy OAI gNB in RF simulator mode and in Standalone Mode

CAUTION: To execute this 2nd step, the whole `CN5G` SHALL be in `healthy` state (especially the `mysql` container).

The gNB can be deployed either in monolithic mode, or in CU/DU split mode.

- For a deployment with the gNB split in CU and DU components:

```
#Deployment of the CU
$ docker-compose up -d oai-cu
#Deployment of the DU
$ docker-compose up -d oai-du
```

Wait for a bit.

```
root@ubuntu:/home/vikash/openairinterface5g/ci-
scripts/yaml_files/5g_f1_rfsimulator# docker-compose ps

       Name                     Command                  State
Ports

-------------------------------------------------------------------------------------------------------
------------------------------

rfsim5g-mysql         docker-entrypoint.sh mysqld      Up (healthy)
3306/tcp, 33060/tcp

rfsim5g-oai-amf       /bin/bash /openair-amf/bin ...   Up (healthy)
38412/sctp, 80/tcp, 9090/tcp

rfsim5g-oai-cu        /opt/oai-gnb/bin/entrypoin ...   Up (healthy)

rfsim5g-oai-du        /opt/oai-gnb/bin/entrypoin ...   Up (healthy)

rfsim5g-oai-ext-dn    /bin/bash -c  apt update;  ...   Up (healthy)

rfsim5g-oai-nrf       /bin/bash /openair-nrf/bin ...   Up (healthy)
80/tcp, 9090/tcp

rfsim5g-oai-smf       /bin/bash /openair-smf/bin ...   Up (healthy)
80/tcp, 8805/udp, 9090/tcp

rfsim5g-oai-spgwu     /openair-spgwu-tiny/bin/en ...   Up (healthy)
2152/udp, 8805/udp
```

You can verify that the gNB is connected with the AMF:

```
$ docker logs rfsim5g-oai-amf
...
[AMF] [amf_app] [info ] |----------------------------------------------------------------------
---gNBs' information----------------------------------------------------------|
[AMF] [amf_app] [info ] |    Index    |      Status       |        Global ID
|       gNB Name       |                PLMN              |
[AMF] [amf_app] [info ] |      1      |    Connected      |          0x0
|         gnb-rfsim        |             208, 99             |
[AMF] [amf_app] [info ] |----------------------------------------------------------------------
-------------------------------------------------------------------------------|
...
```

4. Deploy OAI NR-UE in RF simulator mode and in Standalone Mode

```
root@ubuntu:/home/vikash/openairinterface5g/ci-
scripts/yaml_files/5g_f1_rfsimulator# docker-compose up -d oai-nr-ue

rfsim5g-oai-nrf is up-to-date

rfsim5g-mysql is up-to-date

rfsim5g-oai-amf is up-to-date

rfsim5g-oai-smf is up-to-date

rfsim5g-oai-spgwu is up-to-date

rfsim5g-oai-ext-dn is up-to-date

rfsim5g-oai-cu is up-to-date

rfsim5g-oai-du is up-to-date

Starting rfsim5g-oai-nr-ue ... done
```

Wait for a bit.

```
root@ubuntu:/home/vikash/openairinterface5g/ci-
scripts/yaml_files/5g_f1_rfsimulator# docker-compose ps

       Name                     Command                  State
Ports

-------------------------------------------------------------------------------------------------------
-------------------------------

rfsim5g-mysql        docker-entrypoint.sh mysqld      Up (healthy)
3306/tcp, 33060/tcp
```

```
rfsim5g-oai-amf       /bin/bash /openair-amf/bin ...    Up (healthy)
38412/sctp, 80/tcp,  9090/tcp

rfsim5g-oai-cu        /opt/oai-gnb/bin/entrypoin ...    Up (healthy)

rfsim5g-oai-du        /opt/oai-gnb/bin/entrypoin ...    Up (healthy)

rfsim5g-oai-ext-dn    /bin/bash -c  apt update;  ...    Up (healthy)

rfsim5g-oai-nr-ue     /opt/oai-nr-ue/bin/entrypo ...    Up (healthy)

rfsim5g-oai-nrf       /bin/bash /openair-nrf/bin ...    Up (healthy)
80/tcp, 9090/tcp

rfsim5g-oai-smf       /bin/bash /openair-smf/bin ...    Up (healthy)
80/tcp, 8805/udp, 9090/tcp

rfsim5g-oai-spgwu     /openair-spgwu-tiny/bin/en ...    Up (healthy)
2152/udp, 8805/udp
```

Making sure the OAI UE is connected:

```
$ docker exec -it rfsim5g-oai-nr-ue /bin/bash
root@bb4d400a832d:/opt/oai-nr-ue# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.71.150  netmask 255.255.255.192  broadcast
192.168.71.191
        ether 02:42:c0:a8:47:89  txqueuelen 0  (Ethernet)
        RX packets 224259  bytes 5821372018 (5.8 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 235916  bytes 7848786376 (7.8 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

oaitun_ue1: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 12.1.1.2  netmask 255.255.255.0  destination 12.1.1.2
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen
500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

5. Deploy Second OAI NR-UE in RF simulator mode and in Standalone Mode

```
Create a entry for new IMSI (208990100001101) in oai_db.sql file
```

```
Refer Section - [Making the NR-UE connect to the core network](#51-making-
the-nr-ue-connect-to-the-core-network)

Create entry for Second UE in docker-compose.yaml file as follows:
    oai-nr-ue2:
        image: oai-nr-ue:develop
        privileged: true
        container_name: rfsim5g-oai-nr-ue2
        environment:
            RFSIMULATOR: 192.168.71.140
            FULL_IMSI: '208990100001101'
            FULL_KEY: 'fec86ba6eb707ed08905757b1bb44b8f'
            OPC: 'C42449363BBAD02B66D16BC975D77CC1'
            DNN: oai
            NSSAI_SST: 1
            NSSAI_SD: 1
            USE_ADDITIONAL_OPTIONS: -E --sa --rfsim -r 106 --numerology 1 -
C 3619200000 --nokrnmod --log_config.global_log_options level,nocolor,time
        depends_on:
            - oai-gnb
        networks:
            public_net:
                ipv4_address: 192.168.71.151
        healthcheck:
            test: /bin/bash -c "pgrep nr-uesoftmodem"
            interval: 10s
            timeout: 5s
            retries: 5

root@ubuntu:/home/vikash/openairinterface5g/ci-
scripts/yaml_files/5g_f1_rfsimulator# docker-compose up -d oai-nr-ue

rfsim5g-oai-nrf is up-to-date

rfsim5g-mysql is up-to-date

rfsim5g-oai-amf is up-to-date

rfsim5g-oai-smf is up-to-date

rfsim5g-oai-spgwu is up-to-date

rfsim5g-oai-ext-dn is up-to-date

rfsim5g-oai-cu is up-to-date

rfsim5g-oai-du is up-to-date

Starting rfsim5g-oai-nr-ue2 ... done
```

Wait for a bit.

```
$ docker-compose ps -a
       Name                     Command                  State
Ports
-------------------------------------------------------------------------------------------------------
------------------------------
```

```
rfsim5g-mysql      docker-entrypoint.sh   mysqld      Up   (healthy)
3306/tcp, 33060/tcp
rfsim5g-oai-amf    /bin/bash  /openair-amf/bin   ...   Up   (healthy)
38412/sctp, 80/tcp, 9090/tcp
rfsim5g-oai-ext-dn  /bin/bash  -c  apt  update;  ...   Up   (healthy)

rfsim5g-oai-cu    /opt/oai-gnb/bin/entrypoin   ...   Up   (healthy)

rfsim5g-oai-du       /opt/oai-gnb/bin/entrypoin ...   Up (healthy)

rfsim5g-oai-nr-ue   /opt/oai-nr-ue/bin/entrypo   ...   Up   (healthy)
rfsim5g-oai-nr-ue2   /opt/oai-nr-ue/bin/entrypo   ...   Up   (healthy)
rfsim5g-oai-nrf    /bin/bash  /openair-nrf/bin   ...   Up   (healthy)
80/tcp, 9090/tcp
rfsim5g-oai-smf    /bin/bash  /openair-smf/bin   ...   Up   (healthy)
80/tcp, 8805/udp, 9090/tcp
rfsim5g-oai-spgwu   /openair-spgwu-tiny/bin/en   ...   Up   (healthy)
2152/udp, 8805/udp
```

Making sure the Second OAI UE is connected:

```
$ docker exec -it rfsim5g-oai-nr-ue2 /bin/bash
root@bb4d400a832d:/opt/oai-nr-ue# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.71.151  netmask 255.255.255.192  broadcast
192.168.71.191
        ether 02:42:c0:a8:47:8a  txqueuelen 0  (Ethernet)
        RX packets 3192021  bytes 67784900946 (67.7 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3397743  bytes 91320004542 (91.3 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

oaitun_ue1: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 12.1.1.3  netmask 255.255.255.0  destination 12.1.1.3
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen
500  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

6. Check traffic

Check your Internet connectivity

```
$ docker exec -it rfsim5g-oai-nr-ue /bin/bash
root@bb4d400a832d:/opt/oai-nr-ue# ping -I oaitun_ue1 -c 10 www.lemonde.fr
PING s2.shared.global.fastly.net (151.101.122.217) from 12.1.1.2
oaitun_ue1: 56(84) bytes of data.
```

```
64 bytes from 151.101.122.217 (151.101.122.217): icmp_seq=1 ttl=53
time=64.5 ms
64 bytes from 151.101.122.217 (151.101.122.217): icmp_seq=2 ttl=53
time=37.0 ms
64 bytes from 151.101.122.217 (151.101.122.217): icmp_seq=3 ttl=53
time=43.2 ms
64 bytes from 151.101.122.217 (151.101.122.217): icmp_seq=4 ttl=53
time=43.2 ms
64 bytes from 151.101.122.217 (151.101.122.217): icmp_seq=5 ttl=53
time=54.3 ms
64 bytes from 151.101.122.217 (151.101.122.217): icmp_seq=6 ttl=53
time=24.0 ms
64 bytes from 151.101.122.217 (151.101.122.217): icmp_seq=7 ttl=53
time=32.5 ms
64 bytes from 151.101.122.217 (151.101.122.217): icmp_seq=8 ttl=53
time=37.0 ms
64 bytes from 151.101.122.217 (151.101.122.217): icmp_seq=9 ttl=53
time=41.2 ms
64 bytes from 151.101.122.217 (151.101.122.217): icmp_seq=10 ttl=53
time=50.3 ms

--- s2.shared.global.fastly.net ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9011ms
rtt min/avg/max/mdev = 24.035/42.765/64.557/10.904 ms
```

If it does not work, certainly you need to modify the DNS values in the docker-compose.

But you can also check with the `ext-dn` container (IP address is `192.168.72.135` in docker-compose)

```
$ docker exec -it rfsim5g-oai-nr-ue /bin/bash
root@bb4d400a832d# ping -I oaitun_ue1 -c 2 192.168.72.135
PING 192.168.72.135 (192.168.72.135) from 12.1.1.2 oaitun_ue1: 56(84) bytes
of data.
64 bytes from 192.168.72.135: icmp_seq=1 ttl=63 time=10.9 ms
64 bytes from 192.168.72.135: icmp_seq=2 ttl=63 time=16.5 ms

--- 192.168.72.135 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 10.939/13.747/16.556/2.810 ms
```

Let now try to check UDP traffic in Downlink.

You will need 2 terminals: one in the NR-UE container, one in the ext-dn container.

Note: Similarly, Second OAI UE Internet connectivity can be checked.

6.1 Start the iperf server inside the NR-UE container

```
$ docker exec -it rfsim5g-oai-nr-ue /bin/bash
root@bb4d400a832d:/opt/oai-nr-ue# iperf -B 12.1.1.2 -u -i 1 -s
------------------------------------------------------------
Server listening on UDP port 5001
Binding to local address 12.1.1.2
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
```

6.2 Start the iperf client inside the ext-dn container

```
$ docker exec -it rfsim5g-oai-ext-dn /bin/bash
root@f239e31a0bd0:/# iperf -c 12.1.1.2 -u -i 1 -t 20 -b 500K
------------------------------------------------------------
Client connecting to 12.1.1.2, UDP port 5001
Sending 1470 byte datagrams, IPG target: 22968.75 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.72.135 port 58800 connected with 12.1.1.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  64.6 KBytes   529 Kbits/sec
[  3]  1.0- 2.0 sec  63.2 KBytes   517 Kbits/sec
[  3]  2.0- 3.0 sec  61.7 KBytes   506 Kbits/sec
[  3]  3.0- 4.0 sec  63.2 KBytes   517 Kbits/sec
[  3]  4.0- 5.0 sec  61.7 KBytes   506 Kbits/sec
[  3]  5.0- 6.0 sec  63.2 KBytes   517 Kbits/sec
[  3]  6.0- 7.0 sec  61.7 KBytes   506 Kbits/sec
[  3]  7.0- 8.0 sec  63.2 KBytes   517 Kbits/sec
[  3]  8.0- 9.0 sec  61.7 KBytes   506 Kbits/sec
[  3]  9.0-10.0 sec  63.2 KBytes   517 Kbits/sec
[  3] 10.0-11.0 sec  61.7 KBytes   506 Kbits/sec
[  3] 11.0-12.0 sec  63.2 KBytes   517 Kbits/sec
[  3] 12.0-13.0 sec  61.7 KBytes   506 Kbits/sec
[  3] 13.0-14.0 sec  63.2 KBytes   517 Kbits/sec
[  3] 14.0-15.0 sec  63.2 KBytes   517 Kbits/sec
[  3] 15.0-16.0 sec  61.7 KBytes   506 Kbits/sec
[  3] 16.0-17.0 sec  63.2 KBytes   517 Kbits/sec
[  3] 17.0-18.0 sec  61.7 KBytes   506 Kbits/sec
[  3] 18.0-19.0 sec  63.2 KBytes   517 Kbits/sec
[  3] 19.0-20.0 sec  61.7 KBytes   506 Kbits/sec
[  3]  0.0-20.0 sec  1.22 MBytes   512 Kbits/sec
[  3] Sent 872 datagrams
[  3] Server Report:
[  3]  0.0-20.0 sec  1.22 MBytes   510 Kbits/sec   0.000 ms    3/  872 (0%)
```

Back on your NR-UE terminal you shall see:

```
[  3] local 12.1.1.2 port 5001 connected with 192.168.72.135 port 58800
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total
Datagrams
[  3]  0.0- 1.0 sec  63.2 KBytes   517 Kbits/sec   1.113 ms    0/   44 (0%)
[  3]  1.0- 2.0 sec  61.7 KBytes   506 Kbits/sec   1.466 ms    0/   43 (0%)
[  3]  2.0- 3.0 sec  63.2 KBytes   517 Kbits/sec   1.770 ms    0/   44 (0%)
[  3]  3.0- 4.0 sec  61.7 KBytes   506 Kbits/sec   1.378 ms    0/   43 (0%)
[  3]  4.0- 5.0 sec  63.2 KBytes   517 Kbits/sec   1.614 ms    0/   44 (0%)
[  3]  5.0- 6.0 sec  63.2 KBytes   517 Kbits/sec   1.427 ms    0/   44 (0%)
[  3]  6.0- 7.0 sec  60.3 KBytes   494 Kbits/sec   1.507 ms    1/   43
(2.3%)
[  3]  7.0- 8.0 sec  63.2 KBytes   517 Kbits/sec   1.409 ms    0/   44 (0%)
[  3]  8.0- 9.0 sec  61.7 KBytes   506 Kbits/sec   1.525 ms    0/   43 (0%)
[  3]  9.0-10.0 sec  63.2 KBytes   517 Kbits/sec   1.393 ms    0/   44 (0%)
[  3] 10.0-11.0 sec  61.7 KBytes   506 Kbits/sec   1.377 ms    0/   43 (0%)
[  3] 11.0-12.0 sec  63.2 KBytes   517 Kbits/sec   1.501 ms    0/   44 (0%)
[  3] 12.0-13.0 sec  61.7 KBytes   506 Kbits/sec   1.788 ms    0/   43 (0%)
[  3] 13.0-14.0 sec  63.2 KBytes   517 Kbits/sec   1.466 ms    0/   44 (0%)
[  3] 14.0-15.0 sec  61.7 KBytes   506 Kbits/sec   1.381 ms    0/   43 (0%)
[  3] 15.0-16.0 sec  61.7 KBytes   506 Kbits/sec   1.417 ms    1/   44
(2.3%)
[  3] 16.0-17.0 sec  61.7 KBytes   506 Kbits/sec   1.569 ms    0/   43 (0%)
```

```
[  3] 17.0-18.0 sec  63.2 KBytes   517 Kbits/sec   1.492 ms    1/   45
(2.2%)
[  3] 18.0-19.0 sec  61.7 KBytes   506 Kbits/sec   1.376 ms    0/   43 (0%)
[  3] 19.0-20.0 sec  61.7 KBytes   506 Kbits/sec   1.589 ms    0/   43 (0%)
[  3]  0.0-20.0 sec  1.22 MBytes   510 Kbits/sec   1.551 ms    3/  872
(0.34%)
```

The `500 Kbits/sec` value may change depending on your CPU power!

7. Un-deployment

```
$ docker-compose down
Stopping rfsim5g-oai-nr-ue2 ... done
Stopping rfsim5g-oai-nr-ue  ... done
Stopping rfsim5g-oai-du     ... done
Stopping rfsim5g-oai-cu     ... done
Stopping rfsim5g-oai-ext-dn ... done
Stopping rfsim5g-oai-spgwu  ... done
Stopping rfsim5g-oai-smf    ... done
Stopping rfsim5g-oai-amf    ... done
Stopping rfsim5g-oai-nrf    ... done
Stopping rfsim5g-mysql      ... done
Removing rfsim5g-oai-nr-ue2 ... done
Removing rfsim5g-oai-nr-ue  ... done
Stopping rfsim5g-oai-du     ... done
Stopping rfsim5g-oai-cu     ... done
Removing rfsim5g-oai-ext-dn ... done
Removing rfsim5g-oai-spgwu  ... done
Removing rfsim5g-oai-smf    ... done
Removing rfsim5g-oai-amf    ... done
Removing rfsim5g-oai-nrf    ... done
Removing rfsim5g-mysql      ... done
Removing network rfsim5g-oai-public-net
Removing network rfsim5g-oai-traffic-net
```

8. Explanations on the configuration in the docker-compose.yaml

8.1. Making the NR-UE connect to the core network

The NR-UE **SHALL** be provisioned in the core network, especially in the `SQL` database and in the `AMF`.

- in AMF section of `docker-compose.yaml` --> `OPERATOR_KEY=c42449363bbad02b66d16bc975d77cc1`
- in NR-UE section --> `OPC: 'C42449363BBAD02B66D16BC975D77CC1'

Both values shall match!

This value is also present in the `oai_db.sql` file:

```
INSERT INTO `users` VALUES
('208990100001100','1','55000000000000',NULL,'PURGED',50,40000000,100000000
,47,0000000000,1,0xfec86ba6eb707ed08905757b1bb44b8f,0,0,0x40,'ebd07771ace86
77a',0xc42449363bbad02b66d16bc975d77cc1);
```

```
INSERT INTO `users` VALUES
('208990100001101','1','55000000000000',NULL,'PURGED',50,40000000,100000000
,47,0000000000,1,0xfec86ba6eb707ed08905757b1bb44b8f,0,0,0x40,'ebd07771ace86
77a',0xc42449363bbad02b66d16bc975d77cc1);
```

As you can see, 2 other values shall match in the NR-UE section of `docker-compose.yaml`:

OAI UE - 1

- `FULL_IMSI: '208990100001100'`
- `FULL_KEY: 'fec86ba6eb707ed08905757b1bb44b8f'`

OAI UE - 2

- `FULL_IMSI: '208990100001101'`
- `FULL_KEY: 'fec86ba6eb707ed08905757b1bb44b8f'`

We are also using a dedicated `oai-smf.conf` for the `SMF` container: the `oai` DNN shall match the one in the NR-UE section of `docker-compose.yaml` (`DNN: oai`).

8.2. Making the gNB connect to the core network

Mainly you need to match the PLMN in `gNB`, `AMF` and `SPGWU` parameters:

- AMF
  - MCC=208
  - MNC=99
  - PLMN_SUPPORT_TAC=0x0001
  - …
- SPGWU
  - MCC=208
  - MNC=99
  - TAC=1
- gNB
  - MCC: '208'
  - MNC: '99'
  - TAC: 1

The `ST` and `SD` values shall also match.